(12)

AD-A231 364

FINAL REPORT FOR

OFFICE OF NAVAL RESEARCH GRANT

N00014-88-K-0552

ENTITLED

INTEGRATED DATA AND CONTROL LEVEL FAULT TOLERANCE TECHNIQUES FOR SIGNAL PROCESSING COMPUTER DESIGN

By

G. Robert Redinbo
Department of Electrical Engineering and Computer Science
University of California, Davis

September 1990

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12/18/90	N00014-88-K-0552 (07/01/88-09/30/90)

4. TITLE AND SUBTITLE

INTEGRATED DATA AND CONTROL LEVEL FAULT TOLERANCE TECHNIQUES FOR SIGNAL PROCESSING COMPUTER DESIGN

5. FUNDING NUMBERS

USN N00014-88-K-0552

6. AUTHOR(S)

G. Robert Redinbo

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of California Davis
Davis, California 95616

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Navel Research, Resident Representative
Univesity of California Berkeley
Richmond Filed Stattion, Richmond, CA 94804-001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

High-speed linear signal processing in digital systems is protected efficiently by algorithmic fault-tolerance employing real block or convolutional codes. The main signal processing operation functions normally while parity samples derived from the input data are compared against corresponding parity associated with the output samples. The parity computations and comparisons providing error detection are performed in parallel with the normal signal processing, guaranteeing no speed degradation. Real block codes are used to protect processing finite length input and output segments whereas real convolutional codes are natural for protecting a continuous input and output stream of samples. Algorithrmic fault-tolerance covers a wide class of errors from vatrious levels in the implementation by concentrating on the numerical integrity between related data samples. The corresponding parity values are compared considering difference thresholds to account for numerical roundoff and quantization effects.

A mean-square error criterion is used in analyzing the parity comparison process. Errors due to temporary and permanent hardware failures as well as numerical (cont.)

14. SUBJECT TERMS	15. NUMBER OF PAGES	16. PRICE CODE
	2	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

roundoff and quanization noise are allowed simultaneously in the main processing system, and the parity calculation and comparison subassemblies. The optimum parity estimator for a given code is determined as the conditional mean predictor for the output parity values. The resulting minimum mean-square error guides the simplification of the parallel parity computational overhead. The algebraic structure of the real code permits greatly reducing the parity values' computational effort. For finite input segment processing, real cyclic codes allow the output parity estimates to be calculated from input parity values only. On the other hand, real convolutional codes which are modified without losing error-detecting performance protect the processing of continuous sample streams.

INTEGRATED DATA AND CONTROL LEVEL FAULT TOLERANCE TECHNIQUES FOR SIGNAL PROCESSING COMPUTER DESIGN

G. Robert Redinbo

ABSTRACT

High-speed linear signal processing in digital systems is protected efficiently by algorithmic fault-tolerance employing real block or convolutional codes. The main signal processing operation functions normally while parity samples derived from the input data are compared against corresponding parity associated with the output samples. The parity computations and comparisons providing error detection are performed in parallel with the normal signal processing, guaranteeing no speed degradation. Real block codes are used to protect processing finite length input and output segments whereas real convolutional codes are natural for protecting a continuous input and output stream of samples. Algorithmic fault-tolerance covers a wide class of errors from various levels in the implementation by concentrating on the numerical integrity between related data samples. The corresponding parity values are compared considering difference thresholds to account for numerical roundoff and quantization effects.

A mean-square error criterion is used in analyzing the parity comparison process. Errors due to temporary and permanent hardware failures as well as numerical roundoff and quantization noise are allowed simultaneously in the main processing system, and the parity calculation and comparison subassemblies. The optimum parity estimator for a given code is determined as the conditional mean predictor for the output parity values. The resulting minimum mean-square error guides the simplification of the parallel parity computational overhead. The algebraic structure of the real code permits greatly reducing the parity values' computational effort. For finite input segment processing, real cyclic codes allow the output parity estimates to be calculated from input parity values only. On the other hand, real convolutional codes which are modified without losing error-detecting performance protect the processing of continuous sample streams.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTEGRATED DATA AND CONTROL LEVEL FAULT TOLERANCE TECHNIQUES FOR SIGNAL PROCESSING COMPUTER DESIGN

G. Robert Redinbo

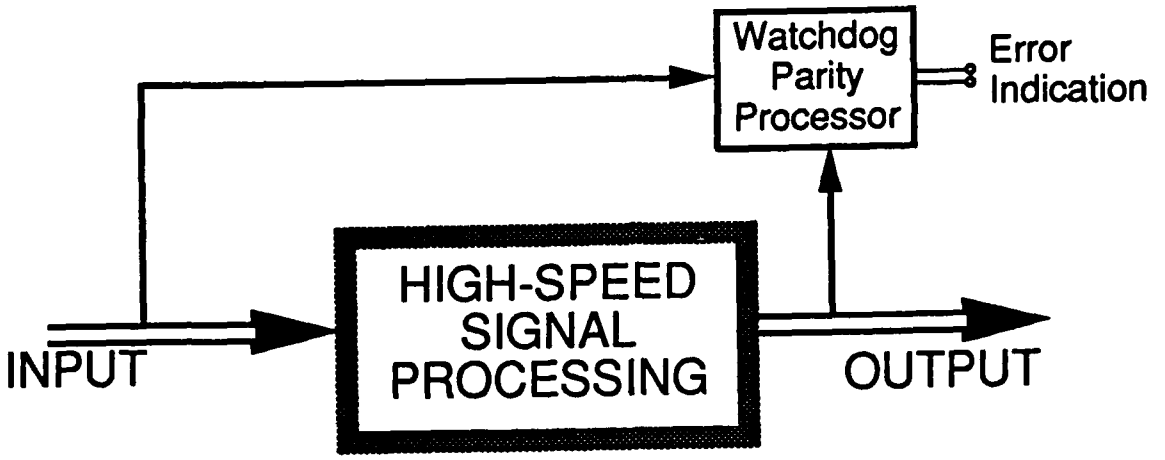
I. INTRODUCTION

High-speed signal processing is an important application of special purpose numerical processor designs. These digital processing systems involve interconnecting complex integrated circuits or a large self-contained VLSI design; both implementations contain dense electronic structures susceptible to internal temporary and permanent failures. Such systems are difficult to protect sufficiently by classical fault-tolerant computer design methods [1] because they employ irregular electronic configurations embedded in otherwise extremely regular structures. Nevertheless, the principal function of signal processing is to manipulate numerical-based data, and the only errors of concern, whether affecting control or data operations, are those corrupting the output numerical data.

The work examined new methods for introducing efficient fault-tolerance design methods in linear signal processing systems without degrading high-speed performance. The main goal was to develop new practical approaches for realistic high speed designs. The fundamental concept relies on a Watchdog Parity Processor that observes both input and output numerical samples and computes a few parity values efficiently (see Figure 1-1). This sparse number of parity values is enough to judge if the main processor is functioning properly in the presence of internal temporary or permanent errors, even in the watchdog unit itself. Only the detection of errors is sufficient for protection since error correction techniques could easily require as many resources as the original processing system. It is more efficient to recompute a series of calculations than to try to correct them.

This approach to fault-tolerant signal processing is representative of a general class of protection techniques called algorithmic fault-tolerance [2-7]. The class derives its name from the fact that certain redundant properties attached to a numerical processing algorithm are used to check the execution and output of the algorithm. The redundancy may be introduced in the basic algorithm using numerical parity codes [2,3,8,9] or may occur naturally in certain forms of an algorithm, e.g., matrix equation solution methods [10].

Algorithmic fault-tolerance protects a potentially very wide class of internal errors. The classical stuck-at type of faults and soft errors are easily covered if they



WATCHDOG PARITY PROCESSOR
FOR NUMERICAL DATA SAMPLE
PROTECTION

FIGURE 1-1

manifest themselves at the numerical level. Numerous errors in a whole subassembly of a design may influence only a few numerical samples where the numerical-based redundancy will detect them. Furthermore, algorithmic fault-tolerance is robust since it is less sensitive to many low-level architectural defects. This approach obtains its requirements from the numerical input-output relationship imposed by the signal processing function.

New protection techniques are now possible due to the relatively recent results concerning real parity codes [7,8,9,11,12]. Both block and convolutional codes have been examined. The block codes can be very powerful generalized cyclic codes which have the best minimum distance structure achievable by a linear code [11,12]. The first class of convolutional codes discussed by Marshall [11] were based on binary codes viewing the binary elements as having counterparts in the real field. However, other classes are possible. It is easy to define a type based on real cyclic codes using an old construction technique of Wyner [13, Section 13.3]. Mathys [14] has another construction for rate $\frac{1}{2}$ real convolutional code based on dual cyclic codes (see Section 11.3 in [15] for theory). Regardless of the exact real code, performance and speed constraints in practical signal processing implementations dictate that any protection method cannot disrupt or alter the original function being protected. This implies that systematic encoding forms of the codes are required [13,15]. This in turn insures that real convolutional codes are noncatastrophic [15].

The next section separates the type of signal processing into two classes depending on the length of the input data segment processed. The linear signal processing is related to a finite or infinite matrix according to the class. The generation of real parity samples is discussed and the necessary notation is established. The third section introduces the role of the mean-square error (MSE) between comparable parity values. Internal hardware errors and roundoff and quantization noise are permitted in the main processor hardware as well as in the subassemblies that compute and compare the parity samples. The optimum parity estimators are developed and the minimum mean-square error expressions, both for finite and infinite processing lengths are given.

The following section presents techniques for simplifying the computational demands in the Watchdog Parity Processor. The efficiencies are based on the algebraic structures of the real parity codes employed. For real cyclic codes, the parity samples attached to finite length convolution are easily calculated from the input parities. On the other hand, when infinitely long input samples are being processed, the real convolutional code's structure may be modified, without degrading the error

protection level, to reduce the parity computation rate. In this way the input data samples are used less frequently in generating the required parity samples.

II. PARITY PROTECTION

The linear signal processing system to be protected may be described by a matrix F relating the input data in a vector $\underline{u}$ to the output data in vector $\underline{y}$. The size of the matrix and respective vectors may be finite or infinite depending on the data being processed.

$$\underline{y} = \underline{u}F \tag{2-1}$$

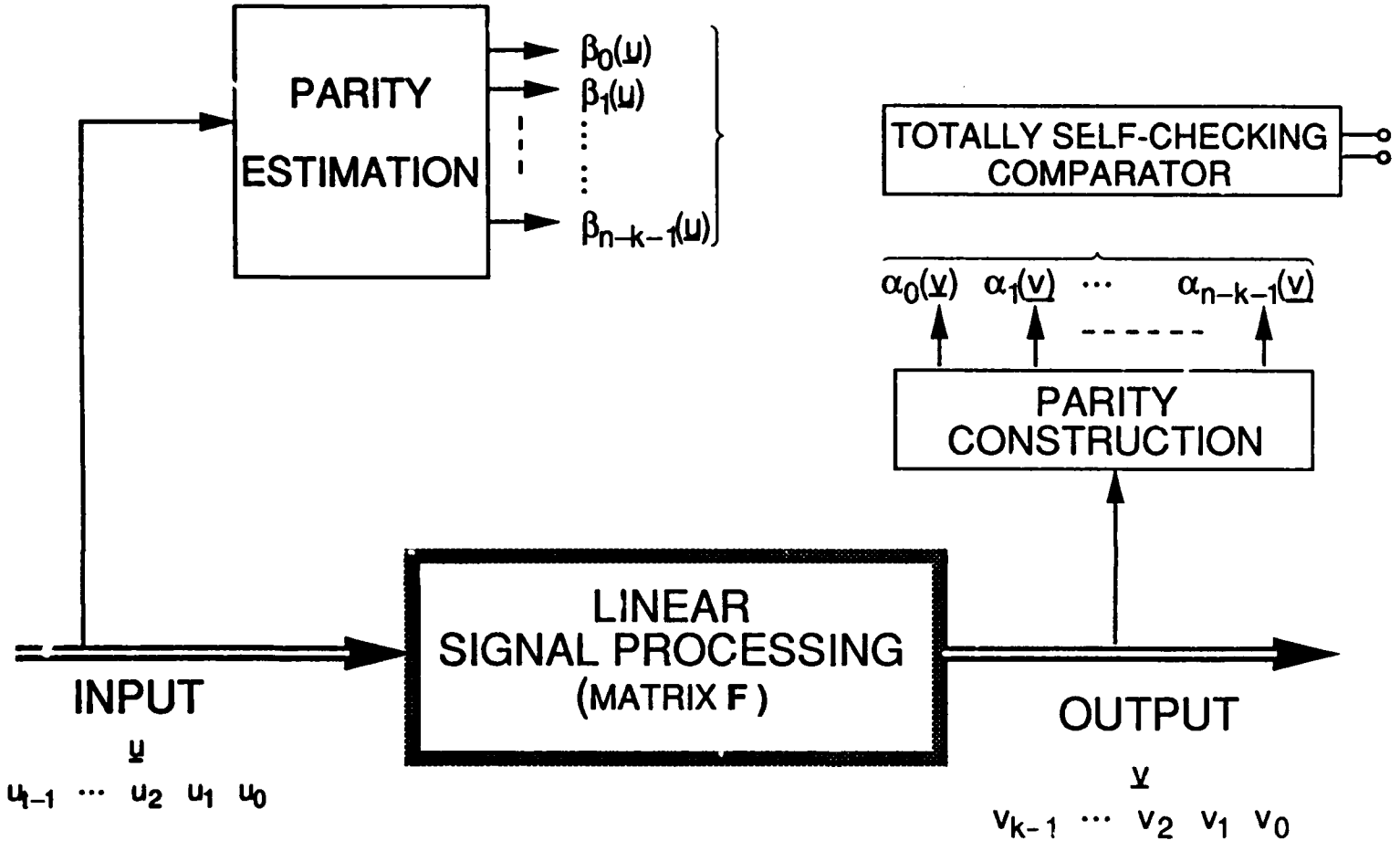
This view encompasses several situations including standard digital filtering using either finite impulse response (FIR) or infinite impulse response (IIR) weighting. Such a model does not constrain the actual implementation details. However, the exact impact of roundoff noise and soft errors on system reliability is directly related to this computational configuration. Infinite arithmetic precision will be assumed, but the effects of roundoff and quantization noise will be included through a probability density function describing the statistical behavior of the model.

Error protection will derive from employing parity samples dictated by real number codes. The existence of powerful real number codes (including maximum distance separable ones) is one reason for choosing this approach. Furthermore, the necessary parity values may be formed in parallel with the normal data processing operations so that there is no speed degradation, a major requirement in fault-tolerant signal processing systems.

A general protection scheme is shown in Figure 2-1. As every group of k input and output samples is processed by the linear system, described by matrix F, two sets of $(n - k)$ parity values are produced respectively at the input and output. Correspondingly related parity samples are checked in a comparator to detect any system errors. In the case where the system processes blocks of data, the code is a real number block (n,k) code [11,12]. On the other hand, if the input is processed as a continuous infinite sequence, a real convolutional code [7,11] is employed. Generally, in either case the number of parity values is kept small to make the protection overhead manageable.

This error detection philosophy derives from experience with finite field based error detection schemes. The parity values are determined by the real code applied to the output samples while their respective estimators are formed in the Parity Estimation subsystem shown in Figure 2-1 where the effects of F and the parity equations are combined.

The distance properties of the real code are used to detect if errors within the implementation have caused any significant differences between the desired and



PROTECTING LINEAR SIGNAL PROCESSING
WITH REAL PARITY VALUES

FIGURE 2-1

actual computation. It is assumed, as is the usual case in fault-tolerant computing system design, that errors occur in only one subassembly of Figure 2-1 at a time: in either the parity estimator, main processing unit, parity constructor, or the comparator. The approach discussed here permits errors in any of there parts, but the heavy concentration of computational effort in the main processing section suggests a higher likelihood of errors there.

The parity estimation part necessarily appears to duplicate the effects of F. However, later results, particularly in important practical cases, will demonstrate how these estimations may be done quite efficiently without doubling the computational effort. Nevertheless, the real number code parities are checked in the comparator. Since roundoff and quantization noise may be present even in the error-free situation, a small comparison threshold must be allowed in determining if two differently calculated parities mismatch due to internal hardware errors.

An accepted method of analyzing roundoff and quantization errors in linear processing systems employs first and second order statistical moments. This analysis basically considers the mean-square average of the difference between ideal results and corresponding values when roundoff and quantization effects are preser.. [16-20]. The mean-square error is prevalent in digital filtering and communication systems [20-22]. The original motivation for error detection uses the minimum probability of error criterion [13,22], but usually roundoff and quantization errors are considered second-order effects and do not influence the design of the error-detecting subsystem. It is natural to view the detection of errors by comparing parity samples as a probability of error motivation, while simultaneously minimizing the mean-square error (MSE) between comparable samples. This dual concept will be explored in the next section where real error-detecting codes with good separation properties will govern the parity construction subassembly while the parity estimation part is designed to minimize the mean-square error between respective parity samples.

The finiteness of the size of the linear processing matrix determines the type of parity code that is employed. For finite length inputs, a systematic real linear block code will be applied, whereas if the input data represent a continuous stream, a systematic real linear convolutional code will be used. The finite processing case with a block code will be examined first, and after some preliminary notation is established, the infinite length case will be addressed. This notation and setting will be the basis for the MSE estimation results developed in the next section.

The linear signal processing operation for finite length data inputs will be represented by the $(k \times k)$ matrix F. The input data are denoted by the $(1 \times k)$ vector $\underline{u}$,

though its tail may actually contain padding zeros because the input is really shorter than k. The output data in vector $\underline{v}$ are obtained through a linear equation. All indexing starts with 0 to be consistent with a concept of time relating to index position.

$$(v_0, v_1, \ldots, v_{k-1}) = \underline{v} = \underline{u}F \;;\; \underline{u} = (u_0, u_1, \ldots, u_{k-1}) \tag{2-2}$$

$$F = \begin{pmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,k-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,k-1} \\ \vdots & \vdots & \vdots & \vdots \\ f_{k-1,0} & f_{k-1,1} & \cdots & f_{k-1,k-1} \end{pmatrix}$$

This more general case includes the special situation of convolving the input data in $\underline{u}$, length t, with a finite impulse response represented by $\underline{w}$, length s, where $s + t \leq k$.

$$\underline{u} = (u_0, u_1, \ldots, u_{t-1}, 0, 0, \ldots, 0)$$

$$\underline{v} = (v_0, v_1, v_2, \ldots, v_{k-1})$$

$$\underline{w} = (w_0, w_1, \ldots, w_{s-1}, 0, 0, \ldots, 0) \Rightarrow F = \begin{pmatrix} w_0 & w_1 & \cdots & w_{s-1} & 0 & \cdots \\ 0 & w_0 & \cdots & w_{s-2} & w_{s-1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & \cdots \end{pmatrix}$$

$$v_r = \sum_{i=0}^{t-1} w_{r-i} u_i \qquad r = 0,1,\ldots,k-1 \tag{2-3}$$

The output data contained in $\underline{v}$ will have parity associated with it. The $(n\text{-}k)$ parity values are determined by a block code whose generator matrix takes the special systematic form.

$$G = (I_k \mid Q) \;;\; I_k \text{ is a } k \times k \text{ identity matrix.} \tag{2-4}$$

The parity-check submatrix,

$$Q = ((q_{ij})) \; ; \quad i=0, 1, \ldots, k-1$$
$$j=0, 1, \ldots, n-k-1.$$

The output's parity values are dictated by the parity-check part of this matrix according to the equation

$$\alpha_j(\underline{v}) = \sum_{i=0}^{k-1} q_{ij} v_i \quad ; \quad j = 0,1,\ldots,n-k-1 \tag{2-5}$$

By design, the real code has good distance separating properties [11], and so any errors appearing in the collection of positions of $\underline{v}$ combined with $\alpha_0(\underline{v})$, $\alpha_1(\underline{v})$, ..., $\alpha_{n-k-1}(\underline{v})$ can be detected. However, section 3 will examine how the parity choices at the output $\underline{v}$ affect the parity estimation subassembly, Figure 2-1.

On the other hand, when the signal processing involves an infinitely long input data stream, the parity samples will be determined by a real systematic convolutional code. One reasonable constraint must be imposed to avoid an infinite delay before any output sample appears. The output v_r in output stream $\underline{v}$ where r=0, 1, ... only depends on input values u_0, u_1, ..., u_r *in the input stream* $\underline{u}$. In other words, the linear signal processing matrix F has zeros below entry r in column r. This causality constraint can be relaxed somewhat by allowing a finite number of nonzero samples beyond item r, but this introduces additional delay that is difficult to implement in practice.

The encoding matrix for a systematic convolutional code, G, has a block type format involving m fundamental finite sized matrices whose dimensions are related to the rate and number of parity check positions in the code. The parameter m determines the constraint length of the code.

$$G = \begin{pmatrix} G_0 & G_1 & — & — & G_m & 0 & — \\ 0 & G_0 & — & — & G_{m-1} & G_m & — \\ 0 & 0 & — & — & — & G_{m-1} & — \\ 0 & 0 & | & | & — & — & | \\ 0 & 0 & — & — & — & — & — \\ | & | & | & G_0 & — & — & | \\ — & — & — & | & G_0 & G_1 & — \\ | & | & | & — & 0 & G_0 & — \\ — & — & — & — & 0 & 0 & — \\ | & | & | & | & | & | & | \end{pmatrix} \tag{2-6}$$

Each $k \times n$ submatrix G_j has a distinctive form.

$$G_0 = (I \mid Q_0) \quad ; \quad \begin{matrix} I & k \times k \text{ Identity Matrix} \\ P_0 & k \times (n-k) \text{ Parity-Check Matrix} \end{matrix} \tag{2-7a}$$

$$G_j = (0 \mid Q_j) \quad ; \quad \begin{matrix} 0 & k \times k \text{ Zero Matrix} \\ P_j & k \times (n-k) \text{ Parity-Check Matrix} \end{matrix} \tag{2-7b}$$

$$j = 1, 2, \ldots, m.$$

The entries in the parity check submatrices Q_j may be either 0 or 1 even for the real Marshal code case [11], or in the more general case, real numbers [7,14].

The parity positions are a function of possibly $(m + 1)k$ input samples through the action of the Q_j parts of each G_j. The stack of these parity weighting values will be denoted by an $\{(m + 1)k \times (n - k)\}$ matrix Q with respective columns $\{ \underline{q_r} \}$:

$$Q = \begin{pmatrix} Q_m \\ Q_{m-1} \\ | \\ | \\ | \\ | \\ Q_2 \\ Q_1 \\ Q_0 \end{pmatrix} = \left(\underline{q_0}, \underline{q_1}, \underline{q_2}, \ldots, \underline{q_{n-k-1}}\right). \tag{2-8a}$$

$$\underline{q_c} = ((q_c^{(j)})) \quad ; \quad j = 0, 1, 2, \ldots, [(m+1)k - 1]. \tag{2-8b}$$

$\underline{q_c}$ $(m+1)k \times 1$ *Column Vector*

$c = 0, 1, 2, \ldots, (n-k-1)$.

The indexing in the parity columns of G is a traditional one which shows parity values as the output of an (FIR) filter. The parity samples are obtained as $(n-k)$ convolution operations on the output data. Employing the notation of equation (2-8), the $(n-k)$ respective parity channels determine the r^{th} parity sample in stream c by the formula:

$$\alpha_c\left(\underline{v}^{(r)}\right) = \sum_{i=0}^{M-1} v_{r-i} q_c^{(i)} \quad ; \quad c = 0,1,\ldots,(n-k-1)$$

$$M = (m+1)k \tag{2-9}$$

where

$\underline{v}^{(r)} = (v_0, v_1, \ldots, v_r)$; output data stream up to item v_r.

III. MINIMUM MEAN-SQUARE ERROR PARITY ESTIMATION

The parity estimation subassembly associated with the real code parities appearing at the processing output will be determined in this section. The case of finite dimensional signal processing is examined first because of its slightly simpler notation. The parity values $\alpha_0(\underline{v}), \alpha_1(\underline{v}), \ldots, \alpha_{n-k-1}(\underline{v})$ are dictated by the real symmetric block code, e.g., equation (2-5). The inherent distance property of this code guarantees that if any reasonably significant errors occur in a number of samples, checking the parity values within error threshold limits will detect them. However, such a detection approach is still true if the parity estimation subassembly presents values whose calculations are directed by a mean-square error criterion. The code is designed for good probability of detection whereas the parity estimation is selected to minimize the mean-square error between related samples. One criterion has gross errors in mind while the MSE focuses on roundoff and quantization effects. Nevertheless, each criterion considers the effects of all errors; there is a different emphasis depending on the type of error.

The general model in Figure 2-1 permits hardware errors as well as roundoff and quantization errors in each of the three computational units: signal processing, parity construction and parity estimation. The mean-square error criterion is applied to the sum of the squared differences between the parity values and their estimated counterparts. This overall criterion is denoted as ε^2.

$$\varepsilon^2 = \sum_{i=0}^{n-k-1} E\left\{\left|\alpha_i(\underline{v}) - \beta_i(\underline{u})\right|^2\right\} \tag{3-1}$$

The expectation uses the distribution governing the elements in the input vector $\underline{u}$ as well as the stationary occurrence of hardware errors and roundoff and quantization noise in the three computational subassemblies.

The role of the totally self-checking comparator is to detect significant differences exceeding the threshold Δ between individual components, including failures and errors within its parts. This is a generalization of the totally self-checking equality checker used in fault-tolerant computing designs [1,23]. One conceptual implementation appears in Figure 3-1a which displays the comparison process for a generic position. The output represents a 1 out of 2 code whenever the two inputs are within a difference magnitude of Δ. Note that this slice is self-testing in that both types the 1 out of 2 code words appear during normal operation [23]. The 1 out of 2 code

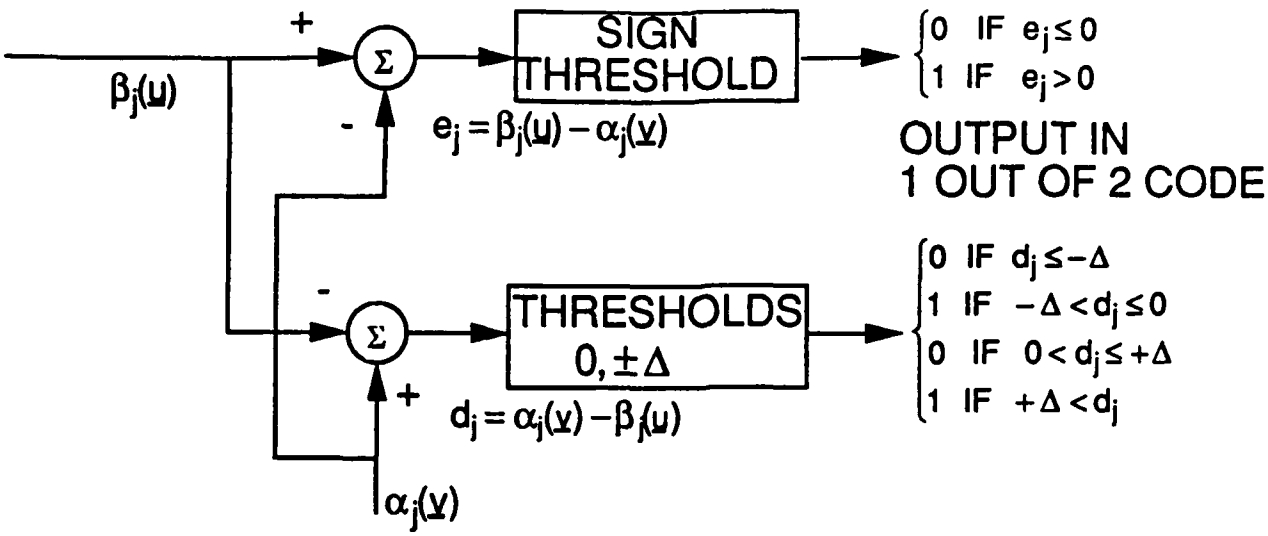


FIGURE 3-1a POSITION COMPARATOR

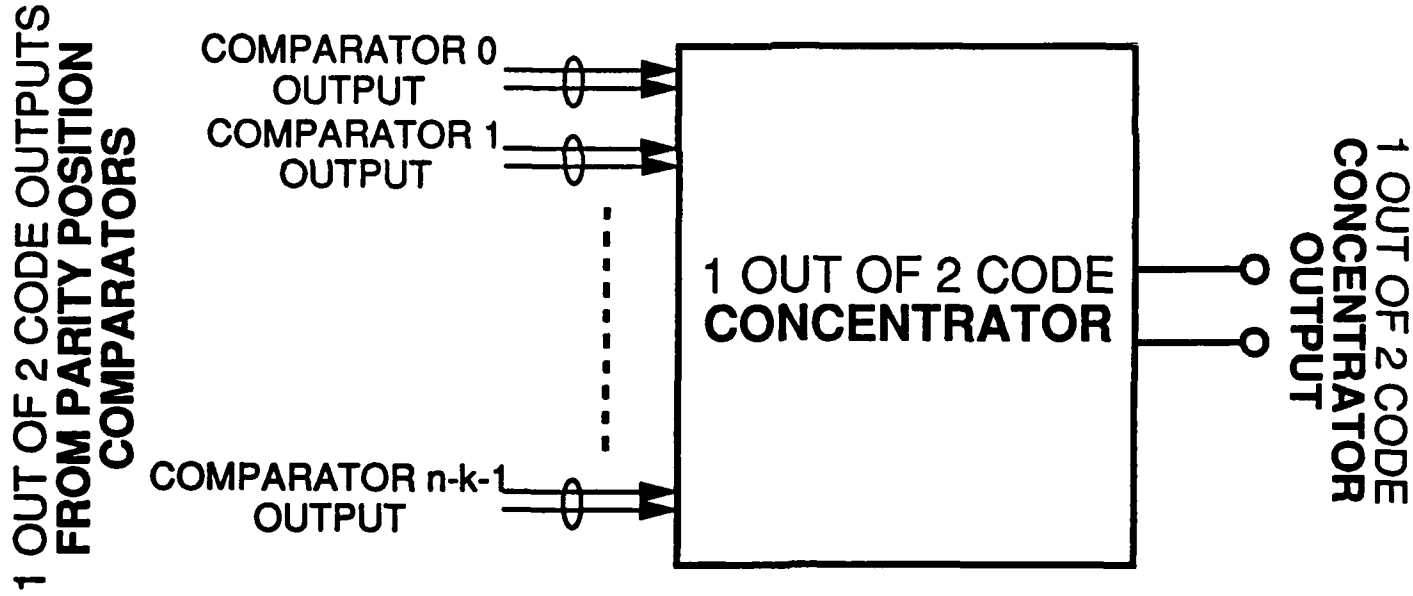


FIGURE 3-1b COMBINING POSITION COMPARATORS

TOTALLY SELF-CHECKING COMPARATOR

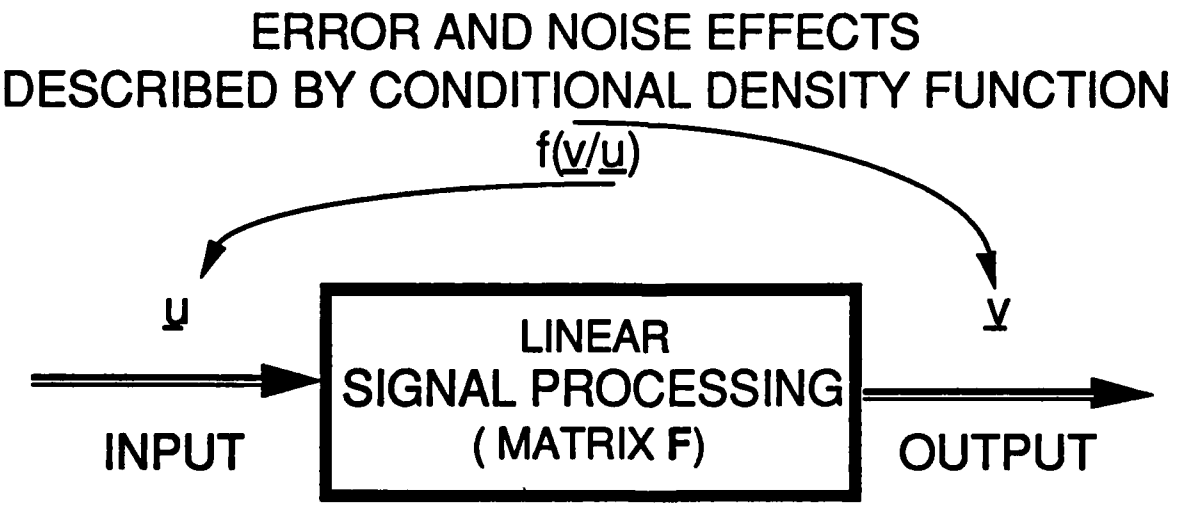
FIGURE 3-1

outputs from each of the $(n - k)$ comparator slices are concentrated to a final 1 out of 2 output as shown in Figure 3-2b. This concentrator is a binary logic device which is code disjoint [1,23] and totally self-checking in the usual sense.

In order to understand the role of the mean-square error criterion on the exact structure of the parity estimator subassembly, a simpler situation will be examined first. All errors, whether hardware or roundoff and quantization errors, will be confined to the signal processing part only. Most of the techniques and mathematical approaches will be expanded later to the situation where hardware errors and roundoff and quantization noise affect all computational parts simultaneously. Furthermore, in this beginning step the linear signal processing subsystem will be taken as finite weighting processing without loss of generality.

Finite Length Processing

The errors in the signal processing part will be described using a conditional density function. (Degenerate forms of errors are incorporated in this model by permitting generalized functions, e.g., Dirac delta functions, in the density function.) This multidimensional conditional density function is denoted by $f_{\underline{v}/\underline{u}}(\underline{v}/\underline{u})$. Since it is



common practice to assume noise in digital signal processing systems can be modeled realistically by additive disturbances [16-19], this conditional density function will be imbued with a generalized additive property. For every $\underline{x}$ in the input space, there is another $1 \times k$ vector $\underline{\xi}(\underline{x})$ such that the conditional density obeys the following law for all input vectors $\underline{u}$ and output vectors $\underline{v}$.

$$f_{\underline{v}/\underline{u}}(\underline{v}/\underline{u}) = f_{\underline{v}/\underline{u}}(\underline{v} + \underline{\xi}(\underline{x})/\underline{u} + \underline{x}) \tag{3-2}$$

The implications of this generalized additivity model can be developed. One candidate for $\underline{x}$ is $-\underline{u}$, and so the effects of errors are typified by passing the all zero vector, denoted by $\underline{0}$, through the signal processing subsystem.

$$f_{\underline{v}/\underline{u}}(\underline{v}/\underline{u}) = f_{\underline{v}/\underline{u}}(\underline{v} + \underline{\xi}(-\underline{u})/\underline{0}) \tag{3-3}$$

The individual vectors $\underline{\xi}(\underline{x})$ as $\underline{x}$ is varied define a transformation from the input space to the output space. It is natural to explore aggregate effects of this transformation. Because property (3-2) holds for all inputs $\underline{u}$ and outputs $\underline{v}$, it is easy to show the following additive properties of the transformation $\underline{\xi}(\underline{x})$.

$$\underline{\xi}(-\underline{u}) = -\underline{\xi}(\underline{u}) \tag{3-4}$$

$$\underline{\xi}(\underline{a} + \underline{b}) = \underline{\xi}(\underline{a}) + \underline{\xi}(\underline{b}) \qquad \underline{a}, \underline{b} \text{ input vectors}$$

Thus the generalized additive assumption leads to a vector transformation $\underline{\xi}$ that has homomorphic properties [24]. Consequently, it is reasonable to infer that the transformation is the signal processing operation.

$$\underline{\xi}(\underline{u}) = \underline{u}F \tag{3-5}$$

The code dictates the parity construction rules $\{\alpha_j(\underline{v})\}$, equation (2-5), while minimizing the mean-square error expression (3-1) will govern the choice of the corresponding parity estimation. It will come as no surprise that the optimum choices for the parity estimator functions are the respective conditional means [20,25].

$$\hat{\beta}_j(\underline{u}) = E_{\underline{v}/\underline{u}}\left[\alpha_j(\underline{v})/\underline{u}\right] \quad j = 0,1,\ldots,(n-k-1) \tag{3-6}$$

The conditional expectations employ the conditional density function $f_{\underline{v}/\underline{u}}(\underline{v}/\underline{u})$ described above. The minimum mean-square error corresponding to conditional mean estimators is given by $\hat{\varepsilon}^2$.

$$\hat{\varepsilon}^2 = \sum_{j=0}^{n-k-1}\left\{E_{\underline{u}}\left[E_{\underline{v}/\underline{u}}\left\{\left|\alpha_j(\underline{v})\right|^2/\underline{u}\right\}\right] - E_{\underline{u}}\left[\left|\hat{\beta}_j(\underline{u})\right|^2\right]\right\} \tag{3-7}$$

A simple proof of these two results are included in a footnote for the sake of completeness.[1]

A vector of optimum parity estimators with respective components given by equation (3-6) may be written employing matrix and vector notation, particularly the parity-check part Q of equation (2-4).

$$\hat{\underline{\beta}}(\underline{u}) = E_{\underline{v}/\underline{u}}\{\underline{v}Q/\underline{u}\} \quad ; \quad Q \text{ is the } k \times (n-k) \text{ parity-check part.} \tag{3-8}$$

The conditional expectation may be simplified by considering the generalized additivity property of the underlying conditional density function, equations (3-4) and (3-5). A vector integral notation will be used to express the conditional expectation.

$$\hat{\underline{\beta}}(\underline{u}) = \left[\int \underline{v} f_{\underline{v}/\underline{u}}(\underline{v}/\underline{u})d\underline{v}\right]Q.$$

A straightforward change of variables, $\underline{z} = \underline{v} - \underline{u}F$, permits this to be rewritten.

---

1 A calculus of variations argument to the proof of the optimality of equations (3-6) and (3-7) above avoids any delicacies that might arise employing a differential approach to minimization. Suppose that there are $(n-k)$ other functions $\gamma_j(\underline{u})$ with finite moments, i.e., $E\{|\gamma_j(\underline{u})|^2\} < +\infty$. The mean-square error resulting from using these functions may be expressed by adding and subtracting the alleged optimum functions $\{\hat{\beta}_j(\underline{u})\}$. The conditional moments may be introduced in this expression.

$$\varepsilon^2 = \sum_{j=0}^{n-k-1} E_{\underline{u}}\left\{E_{\underline{v}/\underline{u}}\left[\left|\alpha_j(\underline{u}) - \hat{\beta}_j(\underline{u}) + \hat{\beta}_j(\underline{u}) - \gamma_j(\underline{u})\right|^2/\underline{u}\right]\right\}$$

$$= \sum_{j=0}^{n-k-1} E_{\underline{u}}\left\{E_{\underline{v}/\underline{u}}\left[\left|\alpha_j(\underline{u}) - \hat{\beta}_j(\underline{u})\right|^2/\underline{u}\right]\right\} + \sum_{j=0}^{n-k-1} E_{\underline{u}}\left\{\left|\hat{\beta}_j(\underline{u}) - \gamma_j(\underline{u})\right|^2\right\}$$

The value $\hat{\varepsilon}^2$, as identified on the right, is a minimum because of the following inequality which becomes strict when $\beta_j(u)$ and $\gamma_j(u)$ differ on any set of nonzero probability.

$$\sum_{j=0}^{n-k-1} E_{\underline{u}}\left\{\left|\hat{\beta}_j(\underline{u}) - \gamma_j(\underline{u})\right|^2\right\} \geq 0.$$

$$\hat{\underline{\beta}}(\underline{u}) = \left[\int \underline{z} f_{\underline{v}/\underline{u}}(\underline{z}/\underline{0})d\underline{z} + \underline{u}F\int f_{\underline{v}/\underline{u}}(\underline{z}/\underline{0})d\underline{z}\right]Q.$$

Thus, in terms of the conditional expectations, the optimum estimator contains a bias term of condition moments of the processing noise.

$$\hat{\underline{\beta}}(\underline{u}) = E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\} + \underline{u}FQ. \tag{3-9}$$

The minimum error $\hat{\varepsilon}^2$, equation (3-7), may be written incorporating the consequences of additivity properties. One part of this expression is the sum of the conditional means.

$$\sum_{j=0}^{n-k-1} E_{\underline{u}}\left\{\left|\beta_j(\underline{u})\right|^2\right\} = E_{\underline{u}}\left\{\underline{u}FQQ^hF^h\underline{u}^h\right\} + E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}QQ^hF^hE_{\underline{u}}\left\{\underline{u}^h\right\}$$

$$+ E_{\underline{u}}\{\underline{u}\}FQQ^hE_{\underline{v}/\underline{u}}\left\{\underline{z}^h/\underline{0}\right\} + E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}QQ^hE_{\underline{v}/\underline{u}}\left\{\underline{z}^h/\underline{0}\right\}. \tag{3-10}$$

The superscript h denotes the conjugate transpose of the vector or matrix. Another important part of $\hat{\varepsilon}^2$ involves the sum of the parity values written as a bilinear form.

$$\sum_{j=0}^{n-k-1}\left|\alpha_j(\underline{v})\right|^2 = \underline{v}QQ^h\underline{v}^h. \tag{3-11}$$

The conditional expectation of this sum may be simplified by using the same change of variables as above, yielding a result using a quadratic form of the vector of the parity values, $\underline{\alpha}(\underline{v})$.

$$E_{\underline{v}/\underline{u}}\left\{\underline{\alpha}(\underline{v})\underline{\alpha}^h(\underline{v})/\underline{u}\right\} = E_{\underline{v}/\underline{u}}\left\{\underline{z}QQ^h\underline{z}^h/\underline{0}\right\} + \underline{u}FQQ^hE_{\underline{v}/\underline{u}}\left\{\underline{z}^h/\underline{0}\right\} + E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}FQQ^h\underline{u}^h. \tag{3-12}$$

Several cancellations occur when combining this equation with the results in equation (3-10) and substituting into equation (3-7) for $\hat{\varepsilon}^2$. In particular, it is interesting to note that the minimum MSE does not involve any averaging over the input distribution.

$$\hat{\varepsilon}^2 = E_{\underline{v}/\underline{u}}\left\{\underline{z}QQ^h\underline{z}^h/\underline{0}\right\} - E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}QQ^hE_{\underline{v}/\underline{u}}\left\{\underline{z}^h/\underline{0}\right\}. \tag{3-13}$$

The complete description of the hardware errors and roundoff and quantization noise is contained in the conditional density function $f_{\underline{v}/\underline{u}}(\underline{v}/\underline{u})$. It may have a mixed type of definition. One realistic example has the hardware errors following a Gaussian density whereas the other noise can have a uniform distribution determined by the smallest level in the numerical representation in the implementation [16,19]. In concrete terms, an example of a modeling density function $f_{\underline{v}/\underline{u}}(\underline{c}/\underline{0})$ may be developed. The roundoff and quantization noise may be lumped together and modeled as k independent components in a vector $\underline{b} = (b_0, b_1, \ldots, b_{k-1})$.

$$f_{\underline{b}}(\underline{b}) = \frac{1}{\delta^k}\prod_{i=0}^{k-1}\mu_{-1}\left(b_i + \frac{\delta}{2}\right)\mu_{-1}\left(\frac{\delta}{2} - b_i\right).$$

The quantization width is given by δ and $\mu_{-1}(x)$ is the unit step function [17]. On the other hand, the hardware errors obey a jointly Gaussian density function where the covariance matrix A and mean value vector $\underline{\rho}$ permit correlation between hardware error events and bias offsets in each component respectively [20].

$$f_{\underline{a}}(\underline{a}) = \left[(2\pi)^{k/2}|A|^{1/2}\right]^{-1}\exp\left\{-\frac{1}{2}(\underline{a}-\underline{\rho})A^{-1}(\underline{a}-\underline{\rho})^T\right\}$$

where

$$\underline{a} = (a_0, a_1, \ldots, a_{k-1}) \qquad \underline{\rho} = (\rho_0, \rho_1, \ldots, \rho_{k-1}) = ((E\{a_i\}))$$

and

$$A = ((E\{(a_i - \rho_i)(a_j - \rho_j)\})) \quad \text{Covariance Matrix.}$$

The combined effects of the hardware errors and performance noise is included as the sum of the random vectors $\underline{a}$ and $\underline{b}$.

$$\underline{c} = \underline{a} + \underline{b}.$$

Assuming that the vectors are mutually statistically independent, the density function of $\underline{c}$ is the k–fold convolution of the two density functions [20,25], written symbolically as:

$$f_{\underline{c}}(\underline{c}) = \int_{\underline{x}} f_{\underline{a}}(\underline{c} - \underline{x}) f_{\underline{b}}(\underline{x}) d\underline{x}.$$

This density function is related to the modeling conditional density through the following requirement:

$$f_{\underline{v}/\underline{u}}(\underline{c}/\underline{0}) = f_{\underline{c}}(\underline{c}) .$$

The bias term, $E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}\underline{Q}$, in the conditional mean estimator $\hat{\underline{\beta}}(\underline{u})$, equation (3-9), does not usually appear in the formulation of the parity estimator. It has generally been ignored in the past. However, omitting it can have an impact on the minimum mean-square error. When this offset term is not included in the parity estimations, it is easy to show that the MSE increases by an amount D,

$$D = E_{\underline{u}}\left\{E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}QQ^{h}E_{\underline{v}/\underline{u}}\left\{\underline{z}^{h}/\underline{0}\right\}\right\} \geq 0. \qquad (3\text{-}14)$$

Hence the parity-check part, Q, of the generator matrix amplifies the effects of any noise mean $E_{\underline{v}/\underline{u}}\{\underline{z}/\underline{0}\}$.

The results discussed so far are too simple. Hardware errors and roundoff and quantization noise can appear in the three major subassemblies. A more realistic simulation will now be detailed. A common assumption in fault-tolerance practice constrains only one subassembly to be affected by hardware failures at a given time. However, roundoff and quantization noise occurs in all three parts simultaneously. The more general model employed below will incorporate all of these features. The goal is to determine the parity estimation functions in such an error environment. The previous results for the simpler case provide important guidance in including the effects of hardware error and performance noise in the estimation subassembly which has not been fully determined yet; a classic chicken-and-egg problem. The key steps in this regard are indicated by the estimator's equations (3-9) for this simple situation, coupled with the basic interconnection of the parts shown in Figure 2-1.

Four probability events will be employed in the error and noise model. They are categorically listed in Table 3-1 along with the symbolic probabilities of their respective occurrences. Events F, E and C represent both hardware errors and roundoff and quantization noise effects, whereas event N accounts for the simultaneous roundoff and quantization noise in the three subassemblies. However, the generalized additive properties of the conditional probability density functions describing all forms of error suggest a realistic tractable modeling technique. The noise effects in the parity estimation part can be reflected through the comparator and considered an additional contribution in the parity construction part. The linear nature of the simple parity

estimators, equation (3-9), coupled with the adder/subtractor in the comparator slices, Figure 3-1a, indicate that extra components of noise added to the parity construction process accurately model the aggregate effects from the estimation subassembly. The exact form of the optimum MSE parity estimation function vector, $\underline{\beta}(\underline{u})$, will be developed using the model. Note the model has roundoff and quantization effects active at all times.

Table 3-1. Error and Noise Events for Subassemblies

Event	Subassembly	Nature of Errors	Probabilities
F	Linear Signal Processing	Soft and Numerical	ρ_F
C	Parity Construction	Soft and Numerical	ρ_C
E	Parity Estimation	Soft and Numerical	ρ_E
N	Processing, Construction and Estimation	Numerical only	ρ_N

$$1 = \rho_F + \rho_C + \rho_E + \rho_N$$

$$\rho_N >> \rho_F + \rho_C + \rho_E$$

The previous development of the mean-square error estimators in equations (3-6) and (3-7) is the starting point in the case of the above expanded model. It is easy to show as earlier that the optimum estimator functions in vector form may be written using the events and probabilities defined in Table 3-1.

$$\hat{\underline{\beta}}(\underline{u}) = \rho_F E\{\underline{\alpha}(\underline{v})|F,\underline{u}\} + \rho_C E\{\underline{\alpha}(\underline{v})|C,\underline{u}\} + \rho_E E\{\underline{\alpha}(\underline{v})|E,\underline{u}\} + \rho_N E\{\underline{\alpha}(\underline{v})|N,\underline{u}\}. \quad (3\text{-}15)$$

Each conditional expectation will be determined individually. The optimum mean-square error expression may be developed as before and can be compactly expressed using vector notation:

$$\hat{\varepsilon}^2 = E_{\underline{u}}\left\{E\left[\underline{\alpha}(\underline{v})\underline{\alpha}^h(\underline{v})/\underline{u}\right] - \hat{\underline{\beta}}(\underline{u})\hat{\underline{\beta}}^h(\underline{u})\right\}. \quad (3\text{-}16)$$

A joint density function involving $\underline{u}$, $\underline{v}$ and $\underline{\alpha}(\underline{v})$ will be useful. It may be decomposed according to the four events detailed in Table 3-1 and contains appropriate conditional density functions.

$$f(\underline{u},\underline{v},\underline{\alpha}) = f_{\underline{u}}(\underline{u})\left[\rho_F f(\underline{v},\underline{\alpha}|F,\underline{u}) + \rho_C f(\underline{v},\underline{\alpha}|C,\underline{u}) + \rho_E f(\underline{v},\underline{\alpha}|E,\underline{u}) + \rho_N f(\underline{v},\underline{\alpha}|N,\underline{u})\right] \quad (3\text{-}17)$$

It must be remembered that the processing output data in $\underline{v}$ and its associated parity $\underline{\alpha}(\underline{v})$ may have a statistical relationship in addition to their deterministic one defined through parity equations (3-11). Hardware errors and roundoff and quantization noise in the parity calculation subassembly introduce this extra dimension.

The first part of the estimator $\underline{\hat{\beta}}(\underline{u})$ depending on event F, where the errors are introduced in the signal processing part, can be developed as before, leading to a result similar to equation (3-9).

$$E\{\underline{\alpha},\underline{v}|F,\underline{u}\} = E_{\underline{v}/F,\underline{v}}\{\underline{z}|F,\underline{0}\} + \underline{u}FQ. \quad (3\text{-}18)$$

On the other hand, event C corresponds to hardware errors and performance noise appearing in the parity construction part. In that case, the pertinent conditional density function has a singular component allowing the following constraint:

$$\begin{aligned} f(\underline{v},\underline{\alpha}|C,\underline{u}) &= f_{\underline{\alpha}|C,\underline{u},\underline{v}}(\underline{\alpha}|C,\underline{u},\underline{v}) f_{\underline{v}|C,\underline{u}}(\underline{v}|C,\underline{u}) \\ &= f_{\underline{\alpha}|C,\underline{u},\underline{v}}(\underline{\alpha}|C,\underline{u},\underline{v}=\underline{u}F) \\ &= f_{\underline{\alpha}|C,\underline{v}}(\underline{\alpha}|C,\underline{v})\Big|_{\underline{v}=\underline{u}F} \end{aligned} \quad (3\text{-}19)$$

Furthermore, the last conditional density function has a generalized additive property.

$$f_{\underline{\alpha}|C,\underline{v}}(\underline{\alpha}|C,\underline{v}) = f_{\underline{\alpha}|C,\underline{v}}(\underline{\alpha} - \underline{v}Q|C,\underline{0}). \quad (3\text{-}20)$$

The component of $\underline{\hat{\beta}}(\underline{u})$ related to event C may be evaluated with the aid of a change of variables, $\underline{y} = \underline{\alpha} - \underline{v}\,Q$.

$$E\{\underline{\alpha}(\underline{v})|C,\underline{u}\} = E_{\underline{\alpha}|C,\underline{v}}\{\underline{y}|C,\underline{0}\} + \underline{u}FQ. \quad (3\text{-}21)$$

The influence of errors in the parity estimation subassembly, event E, is handled the same way except that the conditional density function modeling errors in the $\underline{\hat{\beta}}(\underline{u})$ calculation is transferred to the parity construction part. The resulting conditional

expectation involves an $\underline{\alpha}(\underline{v})$ item, but the underlying density function $f_{\underline{\alpha}\,|\,E,\underline{v}}(\underline{\alpha}\,|\,E,\underline{v})$ represents the errors in the estimation part.

$$E\{\underline{\alpha}(\underline{v})|E,\underline{u}\} = E_{\underline{\alpha}|E,\underline{v}}\{\underline{x}|E,\underline{0}\} + \underline{u}FQ \tag{3-22}$$

For the case of event N where roundoff and quantization effects occur in all three subassemblies, the influence in the parity estimation part is modeled by adding appropriate statistical contributions in the parity construction subassembly. Then the conditional density function describing the filtering and parity construction parts may be separated using Bayes' Rule [20].

$$f(\underline{v},\underline{\alpha}|N,\underline{u}) = f_{\underline{v}|N,\underline{u}}(\underline{v}|N,\underline{u}) f_{\underline{\alpha}|N,\underline{u},\underline{v}}(\underline{\alpha}|N,\underline{u},\underline{v}). \tag{3-23}$$

The first conditional density on the right represents noise in the processing part whereas the second item represents the combined noises in the parity construction subassembly. The nature of the signal flows shows that the last conditional density is not dependent on the input data in $\underline{u}$.

$$f_{\underline{\alpha}|N,\underline{u},\underline{v}}(\underline{\alpha}|N,\underline{u},\underline{v}) = f_{\underline{\alpha}|N,\underline{v}}(\underline{\alpha}|N,\underline{v}). \tag{3-24}$$

Furthermore, two generalized additivity properties come into play:

$$f_{\underline{v}|N,\underline{u}}(\underline{v}|N,\underline{u}) = f_{\underline{v}|N,\underline{u}}(\underline{v} - \underline{u}F|N,\underline{0}) \tag{3-25a}$$

$$f_{\underline{\alpha}|N,\underline{v}}(\underline{\alpha}|N,\underline{v}) = f_{\underline{\alpha}|N,\underline{v}}(\underline{\alpha} - \underline{v}Q|N,\underline{0}) \tag{3-25b}$$

The conditional expectation of interest involves a double integral and two changes of variables, given by $\underline{z} = \underline{v} - \underline{u}F$ and $\underline{y} = \underline{\alpha} - \underline{v}Q$, produce three parts to the conditional expectation,

$$E\{\underline{\alpha}(\underline{v})|N,\underline{u}\} = E_{\underline{\alpha}|N,\underline{v}}\{\underline{y}|N,\underline{0}\} + E_{\underline{v}|N,\underline{u}}\{\underline{z}|N,\underline{0}\}Q + \underline{u}FQ. \tag{3-26}$$

The complete expression for $\hat{\underline{\beta}}(\underline{u})$ combines items from equations (3-18), (3-21), (3-22) and (3-26). The term $\underline{u}FQ$ appears without any event probability weighting because $\rho_F + \rho_C + \rho_E + \rho_N = 1$. A complete bias term which does not depend on input data $\underline{u}$ is given by a vector $\underline{\Gamma}$.

$$\underline{\beta}(\underline{u}) = \underline{u}FQ + \underline{\Gamma}, \tag{3-27}$$

where

$$\underline{\Gamma} = \rho_F E_{\underline{v}|F,\underline{u}}\{\underline{z}|F,\underline{0}\}Q + \rho_C E_{\underline{\alpha}|C,\underline{v}}\{\underline{y}|C,\underline{0}\} + \rho_E E_{\underline{\alpha}|F,\underline{v}}\{\underline{z}|E,\underline{0}\}$$
$$+ \rho_N E_{\underline{\alpha}|N,\underline{v}}\{\underline{y}|N,\underline{0}\} + \rho_N E_{\underline{v}|N,\underline{u}}\{\underline{z}|N,\underline{0}\}Q \tag{3-28}$$

The $1 \times (n - k)$ bias vector $\underline{\Gamma}$ accounts for any nonzero noise averages both due to hardware errors and roundoff and quantization considerations. This vector will appear in the minimum MSE $\hat{\varepsilon}^2$.

The other contributing factor to the MSE, $E\{\underline{\alpha}(\underline{v})\,\underline{\alpha}^h(\underline{v}) \mid \underline{u}\}$, may be expanded in a sum of conditional expectations according to the four events detailed in Table 3-1. These individual conditional expectations may be developed similar to ones above, except that the arguments involve an inner product type expression $\underline{\alpha}(\underline{v})\,\underline{\alpha}^h(\underline{v})$. However, the manipulations and simplifications applied earlier are still pertinent. This development is straightforward but tedious. The final minimum MSE may be assembled according to equation (3-16) and careful bookkeeping shows that all terms containing $\underline{u}$ or $\underline{u}^h$ cancel. Many of the remaining items contain correlations between noise variables.

$$\hat{\varepsilon}^2 = \rho_F E_{\underline{v}|F,\underline{u}}\left\{\underline{z}QQ^h\underline{z}^h|F,\underline{u}=\underline{0}\right\} + \rho_C E_{\underline{\alpha}|C,\underline{v}}\left\{\underline{y}\underline{y}^h|C,\underline{v}=\underline{0}\right\} + \rho_E E_{\underline{\alpha}|E,\underline{v}}\left\{\underline{x}\underline{x}^h|E,\underline{v}=\underline{0}\right\}$$
$$+ \rho_N E_{\underline{\alpha}|N,\underline{v}}\left\{\underline{y}\underline{y}^h|N,\underline{v}=\underline{0}\right\} + \rho_N E_{\underline{\alpha}|N,\underline{v}}\left\{\underline{y}\,|N,\underline{v}=\underline{0}\right\}Q^h E_{\underline{v}|N,\underline{u}}\left\{\underline{z}^h|N,\underline{y}=\underline{0}\right\} \tag{3-29}$$
$$+ \rho_N E_{\underline{v}|N,\underline{u}}\{\underline{z}\,|N,\underline{u}=\underline{0}\}Q\,E_{\underline{\alpha}|N,\underline{v}}\left\{\underline{y}^h|N,\underline{v}=\underline{0}\right\} + \rho_N E_{\underline{v}|N,\underline{u}}\left\{\underline{z}QQ^h\underline{z}^h|N,\underline{u}=\underline{0}\right\} - \underline{\Gamma}\underline{\Gamma}^h.$$

The results for the noise model used here indicate the form of $\hat{\underline{\beta}}(\underline{u})$, equation (3-27), would introduce the same type of noise as the parity construction subassembly. This justifies the reflection of the noise contribution from the estimation part to the construction subassembly.

Infinite Input Data Stream

The case where the linear signal processing handles an infinite input data stream will be protected through parities from a real convolutional code. The general

situation will be narrowed to a more practically important setting. Nevertheless, it will be obvious from the development that more general results, similar to those for the finite length processing case, can be determined in a straightforward manner. The first of two constraints assumes the signal processor is an IIR filter with a rational Z transform. The corresponding matrix F has the infinite impulse response $\{h_r\}_{r=0}^{+\infty}$ as columns.

$$F = \begin{pmatrix} h_0 & h_1 & \cdots & h_r & \cdots \\ 0 & h_0 & \cdots & h_{r-1} & \cdots \\ 0 & 0 & & \vdots & \vdots \\ 0 & 0 & & \vdots & \vdots \\ \vdots & \vdots & & h_1 & \\ \vdots & \vdots & & h_0 & \cdots \\ \vdots & \vdots & & 0 & \cdots \\ \vdots & \vdots & & 0 & \cdots \\ \vdots & \vdots & & \vdots & \end{pmatrix} \text{ IIR filter matrix} \qquad (3\text{-}30)$$

Most practical weighting systems are based on such types of designs [16-19].

The other simplifying constraint concerns the real convolutional code that determines the parity values. A high rate code with only one parity channel will be employed. This means that the information positions are grouped by $k = n - 1$ samples, giving a code rate of $[(n-1)/n]$. It is possible to relax this constraint providing more code alternatives, but for the moment this approach is notationally convenient. The corresponding systematic real convolutional code, described earlier in equations (2-6) through (2-8), employs a parity channel weighting matrix Q with a simplified notation.

$$Q = \begin{pmatrix} q^{(M-1)} \\ q^{(M-2)} \\ \vdots \\ q^{(1)} \\ q^{(0)} \end{pmatrix} \; ; \; M = (m+1)(n-1). \qquad (3\text{-}31)$$

The parity construction part of the system produces samples for this one parity channel according to the following formula:

$$\alpha\left(\underline{v}^{(r)}\right) = \sum_{i=0}^{M-1} v_{r-i} q^{(i)} \quad ; \quad r \geq 0. \tag{3-32}$$

The truncated output data stream denoted by $\underline{v}^{(r)}$ is given notationally as

$$\underline{v}^{(r)} = (v_0, v_1, \ldots, v_r) \quad ; \quad r \geq 0. \tag{3-33}$$

Only the last M samples in this data vector affect the parity construction, but the corresponding input data samples from $\underline{u}$ contributing to $\underline{v}^{(r)}$ are represented by a similar truncated vector.

$$\underline{u}^{(r)} = (u_0, u_1, \ldots, u_r) \quad ; \quad r \geq 0. \tag{3-34}$$

The outputs from the signal processing operations arise from the usual convolution expression.

$$v_t = \sum_{i=0}^{t} h_i u_{t-i} = \sum_{j=0}^{t} h_{t-j} u_j \quad ; \quad t = 0, 1, \ldots \tag{3-35}$$

On the other hand, the parity estimation subassembly provides the parity stream denoted by $\left\{\beta_t\left(\underline{u}^{(t)}\right)\right\}_{t=0}^{+\infty}$. It minimizes the mean-square error between corresponding parity values. The MSE is denoted by ε_t^2 where $t = 0, 1, \ldots$

$$\varepsilon_t^2 = E\left\{\left|\alpha_t\left(\underline{v}^{(t)}\right) - \beta_t\left(\underline{u}^{(t)}\right)\right|^2\right\} \tag{3-36}$$

It may be shown as before that the estimator minimizing this symbol-by-symbol criterion is the conditional mean of the parity construction variable $\alpha_t\left(\underline{v}^{(t)}\right)$.

$$\hat{\beta}_t\left(\underline{u}^{(t)}\right) = E\left\{\alpha_t\left(\underline{v}^{(t)}\right) / \underline{u}^{(t)}\right\} \tag{3-37}$$

The proof supporting this and the development of the related mean-square error, $\hat{\varepsilon}_t^2$, follows the techniques used earlier.

$$\hat{\varepsilon}_t^2 = E\left\{\left|\alpha_t\left(\underline{v}^{(t)}\right)\right|^2\right\} - E_{\underline{u}^{(t)}}\left\{\left|\hat{\beta}_t\left(\underline{u}^{(t)}\right)\right|^2\right\} \tag{3-38}$$

When the only source of errors, both hardware and roundoff and quantization effects, is the signal processing subassembly, the parity estimator can be developed paralleling the previous analysis. The errors are described by a conditional density function which is assumed to obey a familiar generalized additivity property. The optimum estimator has the following form.

$$\hat{\beta}_t\left(\underline{u}^{(t)}\right) = \sum_{i=0}^{M-1} q^{(i)} E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-i}/\underline{u}^{(t)} = \underline{0}\right\} + \sum_{i=0}^{M-1} q^{(i)} \sum_{j=0}^{t-i} h_j u_{t-i-j}. \tag{3-39}$$

This estimator contains the parity of the filtered input as does an expected value needed in ε_t^2.

$$E\left\{\left|\alpha_t\left(\underline{v}^{(t)}\right)\right|^2\right\} = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} q^{(i)} q^{(j)*} \left\{E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-i} z_{t-i}^*/\underline{0}\right\}\right.$$

$$+ E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-i}/\underline{0}\right\} \sum_{r=0}^{t-j} h_r^* E\left\{u_{t-j-r}^*\right\} + E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-j}^*/\underline{0}\right\} \sum_{r=0}^{t-i} h_r E\left\{u_{t-i-r}\right\} \tag{3-40}$$

$$\left. + \sum_{r=0}^{t-i} \sum_{s=0}^{t-j} h_r h_s^* E\left\{u_{t-i-r} u_{t-j-s}^*\right\}\right\}$$

Not surprisingly, the minimum MSE is only dependent on the finite length of data needed in the parity construction subassembly even though samples for the entire length of $\underline{u}^{(t)}$ and $\underline{v}^{(t)}$ appear in the two previous expressions.

$$\hat{\varepsilon}_t^2 = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} q^{(i)} q^{(j)*} \left[E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-i} z_{t-j}^*/\underline{0}\right\} - E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-i}/\underline{0}\right\} E_{\underline{v}^{(t)}/\underline{u}^{(t)}}\left\{z_{t-j}^*/\underline{0}\right\}\right] \tag{3-41}$$

The correlation between noise samples in the processing subassembly plays a prominent role just as in the finite length processing case, e.g., equation (3-13).

The situation where hardware errors are permitted in any one of the three subassemblies while roundoff and quantization disturbances are present in all parts can be addressed employing the same error model described in detail earlier. The nature and type of error events remain as given in Table 3-1. A set of familiar results emerges. The parity estimation part contains a bias term in addition to the expected parity affiliated with the processing part's output.

$$\hat{\beta}_t\left(\underline{u}^{(t)}\right) = \sum_{i=0}^{M-1} q^{(i)} \sum_{j=0}^{t-i} h_j\, u_{t-i-j} + \Lambda_t \tag{3-42a}$$

where

$$\Lambda_t = \rho_F \sum_{i=0}^{M=1} q^{(i)} E_{\underline{v}^{(t)}/F,\underline{u}^{(t)}}\{z_{t-i}/F,\underline{0}\} + \rho_N \sum_{i=0}^{M=1} q^{(i)} E_{\underline{v}^{(t)}/N,\underline{u}^{(t)}}\{z_{t-i}/N,\underline{0}\}$$

$$+ \rho_N E_{\alpha^{(t)}/N,\underline{v}^{(t)}}\{y_t/N,\underline{0}\} + \rho_C E_{\alpha^{(t)}/C,\underline{v}^{(t)}}\{y_t/C,\underline{0}\} + \rho_E E_{\alpha^{(t)}/E,\underline{v}^{(t)}}\{y_t/E,\underline{0}\} \tag{3-42b}$$

and

$$\hat{\varepsilon}_t^2 = \rho_N \sum_{i=0}^{M=1} \sum_{j=0}^{M=1} q^{(i)} q^{(j)^*} E_{\underline{v}^{(t)}/N,\underline{u}^{(t)}}\left\{z_{t-i}\, z_{t-j}^*/N,\underline{0}\right\}$$

$$+ \rho_F \sum_{i=0}^{M=1} \sum_{j=0}^{M=1} q^{(i)} q^{(j)^*} E_{\underline{v}^{(t)}/F,\underline{u}^{(t)}}\left\{z_{t-i}\, z_{t-j}^*/F,\underline{0}\right\}$$

$$+ \rho_C E_{\alpha^{(t)}/C,\underline{v}^{(t)}}\left\{|y_t|^2/C,\underline{0}\right\} + \rho_E E_{\alpha^{(t)}/C,\underline{v}^{(t)}}\left\{|y_t|^2/E,\underline{0}\right\} + \rho_N E_{\alpha^{(t)}/N,\underline{v}^{(t)}}\left\{|y_t|^2/N,\underline{0}\right\} \tag{3-43}$$

$$+ \rho_N \sum_{i=0}^{M-1} q^{(i)^*} E_{\underline{v}^{(t)}/N,\underline{u}^{(t)}}\left\{z_{t-i}^*/N,\underline{0}\right\} E_{\alpha^{(t)}/N,\underline{v}^{(t)}}\{y_t/N,\underline{0}\}$$

$$+ \rho_N \sum_{i=0}^{M-1} q^{(i)} E_{\underline{v}^{(t)}/N,\underline{u}^{(t)}}\left\{z_{t-i}/N,\underline{0}\right\} E_{\alpha^{(t)}/N,\underline{v}^{(t)}}\left\{y_t^*/N,\underline{0}\right\} - |\Lambda_t|^2.$$

It is easy to identify the sources of the noise effects in both the estimators $\hat{\beta}_t\left(\underline{u}^{(t)}\right)$ and the minimum MSE value $\hat{\varepsilon}_t^2$. Those items containing conditional expectations with subscripts of the form $\underline{v}^{(t)}/A,\underline{u}^{(t)}$ (where A is a generic letter) represent noises in the signal processing part. On the other hands, those conditional expectations having subscripts of the form $\alpha^{(t)}/A,\underline{v}^{(t)}$ designate noise sources in either the parity calculation part or parity estimation subassembly as reflected through to the construction part.

IV. EFFICIENT USE OF REAL CODES

The optimum mean-square error parity estimator, whether for finite or infinite length signal processing operations, contains a term representing the parity of the processed output. For finite length data processing, the term $\underline{\mu}FQ$ contains the processed output $\underline{\mu}F$ (equations (3-9) or (3-27)). On the other hand, when processing a continuous infinite input data stream the double sum

$$\sum_{i=0}^{M-1} q^{(i)} \left[\sum_{j=0}^{t-i} h_j \, u_{t-i-j} \right]$$

is the parity for M outputs of the processing subassembly (see equations (3-37) or (3-42)). At first glance it appears that the parity estimator must duplicate the complete processing effort of the signal processing subassembly, implying that the protection overhead must exceed 100%. Thus the efficiency of the parity estimator subassembly is an important factor in offering any practical advantages. Fortunately, for both finite and infinite length processing systems, techniques exist for dramatically reducing the computational requirements in the parity construction and estimation subassemblies.

An effective way to reduce parity computation complexities is to match the processing calculation more closely to the structure of the real code. Two techniques will be discussed. In the first approach, close similarity between finite convolution and real cyclic codes reduces the parity estimation calculations to the order of the number of parity values squared. On the other hand, the parity channels used to protect infinite impulse response filtering can be modified so as to cancel the poles of the original transfer function. This not only removes the need for feedback memory requirements in the parity estimation part, but more importantly, permits the parity construction and estimation subassemblies to operate at lower computational rate. This in turn offers very efficient implementation options. The details of both approaches will be given below, starting with protecting finite length convolutions.

Protection With Real Cyclic Codes

Finite length convolutions may be described compactly in terms of polynomials [26]. The filter weighting sequence may be viewed as a polynomial $f(X)$ whose coefficients are related to the signal processing matrix in the following way.

$$F = \begin{pmatrix} f_0 & f_1 & f_2 & \cdots & f_s & \cdots \\ 0 & f_0 & f_1 & \cdots & f_{s-1} & \cdots \\ 0 & 0 & f_0 & \cdots & \vdots & \cdots \\ \vdots & 0 & 0 & \cdots & \vdots & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ 0 & 0 & 0 & \cdots & f_1 & \cdots \\ \vdots & \vdots & \vdots & \cdots & f_0 & \cdots \\ \vdots & \vdots & \vdots & \cdots & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \vdots & \cdots \end{pmatrix} \quad k \times k \text{ Convolution Matrix, } s \le k \qquad (4\text{-}1a)$$

$$f(X) = f_s X^s + f_{s-1} X^{s-1} + \ldots + f_1 X + f_0 \quad ; \quad \text{Weighting Polynomial} \qquad (4\text{-}1b)$$

When the input data are represented in polynomial form, $u(X)$, the convolution between the input data and the weighting sequence, $f(X)$, is a polynomial product $f(X)\,u(X)$. If the integer k is chosen large enough (deg $f(X)$ + deg $u(X) < k$), the convolution weighting may be viewed in a residue polynomial ring modulo $(X^k - 1)$ [24,26] where the output convolution is contained in polynomial $v(X)$,

$$v(X) = u(X)\,f(X) \text{ modulo } (X^k - 1)\,. \qquad (4\text{-}2)$$

Real cyclic codes are defined in a similar residue polynomial ring where the modulo is performed $(X^n - 1)$, $n > k$. The code has $(n - k)$ parity positions and length n. It is defined by a single polynomial, called the generator polynomial and designated here as $g(X)$. This polynomial has degree $(n - k)$, may be selected with a unity leading coefficient, and divides $(X^n - 1)$. The code is all the polynomials in a principal ideal [24] generated by $g(X)$. This ideal is denoted by double parentheses around $g(X)$, (($g(X)$)), and is formally defined as the set of polynomials in the residue polynomial ring that are multiples of $g(X)$:

$$((g(X))) = \{p(X) \equiv q(X)\,g(X) \text{ modulo } (X^{n-1}) \text{: } q(X) \text{ a polynomial}\} \qquad (4\text{-}3)$$

A common method for defining a systematic form of a cyclic code employs the famous Euclidean algorithm [24,26] which will be stated here for completeness. Any polynomial $f(X)$ may be divided by $g(X)$ and expressed uniquely in terms of a quotient $q(X)$ and a remainder $r(X)$, with an important constraint on the remainder's degree:

$$f(X) = q(X)\,g(X) + r(X) \quad ; \quad \deg r(X) < \deg g(X)\,. \tag{4-4}$$

Cyclic codes over the real or complex fields are defined by using consecutively indexed powers of the n^{th} complex root of unity, e.g., $[\exp(j2\pi/n)p]$, where $j = \sqrt{-1}$ and p is an integer. The fundamental construction techniques are given by Marshall [11,27], and use the discrete Fourier transform domain in which contiguously indexed transform coefficients determine the generator polynomial. By requiring conjugate roots be included, real generator polynomials are constructed. Maximum distance separable codes [11,13] (analogous to powerful Reed-Solomon codes) are easily constructed. It was established in [27] that real number maximum distance separable codes exist for all choices of parameters. Such codes can detect errors equal in number to the degree of $g(X)$, the maximum permitted by the Singleton bound [13].

Real cyclic codes, primarily applicable to floating point arithmetic formats, will be examined in slightly more detail so as to better exemplify the parity operations to be discussed later. They are constructed using consecutively indexed primitive roots of unity [11] and were originally called Discrete Fourier Transform (DFT) codes [11,12,27,28]. Powers of the n^{th} complex root of unity, W, define the roots of the code generator polynomial $g(X)$.

$$g(X) = \prod_{\xi \in \Xi} \left(X - W^{\xi}\right): \qquad W = e^{\frac{j2\pi}{n}} \tag{4-5}$$

Ξ = INDEX SET OF CONSECUTIVE INTEGERS MOD n

$$\Xi = \{0, 1, 2, \ldots, (n-1)\}$$

The span (number of consecutive indices) determines the error-detecting capability of the code and is the maximum allowable for a linear code. The code can detect up to $|\Xi|$ symbol positions in error considering the roundoff tolerance of the comparison operation.

If the index root set is symmetric about 0 (and including 0) or about $n/2$ (and including $n/2$ if n is even), conjugate root pairs appear in $g(X)$, giving it real coefficients. This restriction narrows the range of parameters permitted in the code slightly (see Property 3 of [11]), but does not reduce the error protection levels in any way. In addition, the real coefficients of $g(X)$ are also symmetric or anti-symmetric

about the degree midpoint, halving the number of multiplicative operations needed in parity calculation and re-calculation. This implementation detail will not be exploited further in the paper to avoid unnecessarily complicating the equations.

A power of the primitive complex root, W^m, where m is relatively prime to n can be used in place of W in the definition of $g(X)$. A different code results with the same error-detecting capability but with roots located at more widely dispersed points on the unit circle. This eases the accuracy requirement in calculating and using the coefficients in $g(X)$. A simple example for a code with length $n = 1024$ and information capacity $k = 1003$ using an index scaling factor $m = 47$, has the capability of detecting up to 21 positions in error or any burst up to length 21. Figure 4-1 displays the generator polynomial. An equally good code may have roots centered about $-1 = W^{512}$, but they have different coefficient signs since $(X+1)$ is the only linear factor.

$$
\begin{aligned}
g(X) = {} & -1.000000e{+}00 \;+7.882733e{-}01X^{1} \;-7.063940e{-}01X^{2} \\
& +6.591348e{-}01X^{3} \;-6.277460e{-}01X^{4} \;+6.055386e{-}01X^{5} \\
& -5.894287e{-}01X^{6} \;+5.777794e{-}01X^{7} \;-5.696514e{-}01X^{8} \\
& +5.644898e{-}01X^{9} \;-5.619806e{-}01X^{10} \;+5.619806e{-}01X^{11} \\
& -5.644898e{-}01X^{12} \;+5.696514e{-}01X^{13} \;-5.777794e{-}01X^{14} \\
& +5.894287e{-}01X^{15} \;-6.055386e{-}01X^{16} \;+6.277460e{-}01X^{17} \\
& -6.591348e{-}01X^{18} \;+7.063940e{-}01X^{19} \;-7.882733e{-}01X^{20} \\
& +1.000000e{+}00X^{21}
\end{aligned}
$$

Generator Polynomial Using Root W^{47}

$n = 1024 \quad ; \quad k = 1003$

A REAL CYCLIC CODE GENERATOR POLYNOMIAL
FIGURE 4-1

A common systematic encoding scheme will be used to describe the efficient calculation of parity values associated with the output convolution. The input data represented by $u(X)$ is encoded in the following format. The data in $u(X)$ are placed in higher coefficient positions by multiplying it by X^{n-k}, effectively shifting this data to

inclusively indexed positions from $(n-k)$ up to $(n-1)$. The uniquely related parity symbols are represented by the polynomial $r_u(X)$, derived from the Euclidean algorithm (4-4) with $g(X)$ as the divisor.

$$X^{n-k}u(X) = q_u(X)\,g(X) + r_u(X) \quad ; \quad \deg r_u(X) < \deg g(X) \tag{4-6}$$

$$\text{ENCODE } u(X) \rightarrow X^{n-k}u(X) - r_u(X)$$

$$\text{PARITY PART} = -r_u(X)$$

The parity estimation subassembly may use the parity $r_u(X)$, which is carried along with the input data $u(X)$, in forming the parity $-r_v(X)$ corresponding to the output $v(X)$, equation (4-2). The formula relating the input parity $-r_u(X)$ to the output parity $r_v(X)$ involves a weighting polynomial $\psi(X)$ which may be computed and stored. Because $\psi(X)$ is the remainder modulo $g(X)$, it has degree strictly less than $(n-k)$.

$$r_v(X) \equiv r_u(X)\,\psi(X) \text{ modulo } g(X) \tag{4-7a}$$

where

$$\psi(X) \equiv f(X) \text{ modulo } g(X)\,. \tag{4-7b}$$

The proof[2] that this is indeed the proper output parity is contained in a footnote which makes use of the properties from the Euclidean algorithm. The computation of the parity estimator uses equation (4-7a) which is much simpler than a straightforward computation. Its complexity is on the order of $(n-k)^2$. Furthermore, if the parity values may be computed and compared in a suitable transform domain, this complexity drops to the order of $(n-k)$ [29].

The systematic generator matrix for a cyclic code may be described by mixing vector and polynomial notation. The I^{th} row of G relates to the cyclic codeword for $\{X^{n-1-I}\}$ where $r = 0, 1, \ldots, k-1$. In particular, when this code polynomial denoted by $\{X^{n-1-r} - p_I(X)\}$ is written as a vector, the X^{n-1-r} terms correspond to the identity part I_k while the parity parity, remainder $- p_I(X)$, is confined to the rightmost $(n-k-1)$ matrix columns.

$$X^{n-1-r} = q_I(X)\, g(X) + p_I(X) \quad ; \quad \deg p_I(X) < n-k \tag{4-8}$$

---

2 PROOF OF SIMPLIFIED PARITY CALCULATION, EQUATION (4-7).

The parity associated with the output $v(X) = u(X)\, f(X)$ modulo $(X^k - 1)$ will be denoted by $\xi(X)$. The Euclidean algorithm shows there is a quotient $q_I(X)$ and a remainder $\psi(X)$ obeying

$$f(X) = q_I(X)\, g(X) + \psi(X)\,.$$

This gives equivalence (4-7b). On the other hand, the product $r_U(X)\, \psi(X)$ when reduced modulo $g(X)$ has remainder denoted by $r_V(X)$.

$$r_U(X)\, \psi(X) = q_p(X)\, g(X) + r_V(X)\,.$$

The validity of equation (4-7a) may be seen by expanding the difference between the shifted output data, $X^{n-k}[u(X)\, f(X)]$, and the parity $r_V(X)$.

$$X^{n-k}[u(X)\, f(X)] - r_V(X) = \{q_U(X)g(X) + r_U(X)\}\, \{q_I(X)g(X) + \psi(X)\} - q_p(X)g(X) + r_U(X)\psi(X)$$

$$= q_U(X)q_I(X)g^2(X) + r_U(X)q_I(X)g(X) + q_U(X)g(X)\psi(X) - q_p(X)g(X)\,.$$

Since $r_V(X)$ has degree less than $(n-k)$ and is unique by its very construction, it must be the corresponding parity because the difference above is a multiple of the code generator polynomial $g(X)$.

In order to be somewhat consistent with the earlier block code notation associated with equation (2-4), the coefficients have a reversed order.

$$p_r(X) = \sum_{i=0}^{n-k-1} p_{r,i}\, X^{n-1-r} \;\; ; \;\; ((\, p_{r,i}\,)) = \mathrm{P}, \;\; r=0,1,\ldots,(k-1). \tag{4-9}$$

Then the systematic generator matrix G in the cyclic code has the following form involving matrix P defined by remainder polynomials $p_r(X)$.

$$\mathrm{G} = (I_k \,|\, \mathrm{P}) = ((\, X^{n-1-r} - p_r(X)\,)) \quad ; \quad r=0,\,1,\,\ldots,\,(k-1). \tag{4-10}$$

The parity construction subassembly implements the following function where the reverse indexing and minus sign appear because of the notational definition of the real cyclic code.

$$\alpha_r(\underline{v}) = -\sum_{j=0}^{k-1} v_j\, p_{k-1-j} \;\; ; \;\; r=0,\,1,\,\ldots,\,(k-1). \tag{4-11}$$

This definition gives parity values as generated by the Euclidean algorithm encoding format, equation (4-6). The encoding of output $v(X)$ has a parity polynomial part $r_v(X)$ that may be expressed using the $\alpha_r(\underline{v})$ values.

$$X^{n-k}v(X) = q_v(X)q(X) + r_v(X) \tag{4-12a}$$

$$r_v(X) = -\left[\sum_{i=0}^{n-k-1} \alpha_i(\underline{v})\, X^i\right]. \tag{4-12b}$$

Typical terms in the optimum MSE value $\hat{\varepsilon}^2$ contain bilinear forms of the parity matrix Q and other statistical quantities, e.g., equations (3-13) or (3-29). Some simplifying assumptions will be made in order to study the impact on MSE performance by choices for the parity coefficients in parity part matrix P, equation (4-9). These parity coefficients depend on the generator polynomial which in turn is defined

by its indexing subject Ξ, equation (4-5). This will be explored further by concentrating on one bilinear form, say item E.

$$E \triangleq E_{\underline{v}/\underline{u}}\left\{\underline{z}QQ^h\underline{z}^h/\underline{0}\right\} = \sum_{r=0}^{n-k-1}\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} p_{k-1-i,r}\, p_{k-1-j,r} E_{\underline{v}/\underline{u}}\left[z_i z_j^*/\underline{0}\right] \qquad (4\text{-}13)$$

This bilinear form contains correlation statistics concerned with errors appearing in a subassembly's error model. As a beginning point, the noise will be taken as uncorrelated so that individual component noise powers, σ_j^2, are the only nonzero statistics.

$$E_{\underline{v}/\underline{u}}\left[z_i z_j^*/\underline{0}\right] = \begin{cases}\sigma_i^2 & i=j \\ 0 & i\neq j\end{cases}. \qquad (4\text{-}14)$$

The bilinear form reduces to a sum of squares.

$$E = \sum_{r=0}^{n-k-1}\sum_{i=0}^{k-1}\sigma_i^2\left|p_{k-1-i,r}\right|^2 \qquad (4\text{-}15)$$

Note that the noise powers σ_j^2 are not necessarily from a wide-sense stationary process [20]. It is easy to show that such a situation arises in a direct realization of finite convolution when scalers and summers are affected by uncorrelated noise. Also the σ_j^2 have a linear relationship for indices in the beginning and end of the indexing range.

Real cyclic codes are defined using n^{th} linear complex roots of unity, and it is natural to introduce a transform in E. It appears that a Parseval's type relationship might apply to the squared items in equation (4-15). Therefore, some notational conventions will be introduced for a discrete Fourier transform [16-19,26-28]. The data sequence will be denoted by $\{a_r\}$, $r = 0, 1, \ldots, (n-1)$, and its transform is designated by the transform sequence $\left\{\hat{A}_q\right\}$, $q = 0, 1, \ldots, (n-1)$.

$$\hat{A}_q = \sum_{r=0}^{n-1} a_r W^{rq} \;;\; q = 0, 1, \ldots, (n-1). \qquad (4\text{-}16a)$$

The usual inverse transform applies:

$$a_r = \frac{1}{n} \sum_{q=0}^{n-1} \hat{A}_q W^{-rq} \quad ; \quad r = 0, 1, \ldots, (n-1) . \tag{4-16b}$$

The transform pair will be used on individual rows of the generator matrix G, equation (4-10). The i^{th} row, n samples long produces a transform sequence $\left\{\hat{P}_{i,q}\right\}$.

$$\hat{P}_{i,q} = W^{q(n-1-i)} - \sum_{r=0}^{n-k-1} p_{i,r} W^{q(n-k-1-r)} \quad ;$$

$$q = 0, 1, \ldots, (n-1) \quad ; \quad i = 0, 1, \ldots, (k-1) . \tag{4-17}$$

The first item is from the monomial X^{n-1-i} representing the identity part of G while the remaining sum arises from the parity polynomial–$p_i(X)$ for this i^{th} row.

The cyclic code's definition designates which powers of W are roots of the rows of G, viewed as polynomials. From equation (4-5), if index $\xi \in \Xi$, the index subset, W^{ξ}, is a root of each $[X^{n-1-i} - p_i(X)]$. This gives an evaluation of $p_i(X)$ at all $(n-k)$ indices in Ξ.

$$p_i\left(W^{\xi}\right) = W^{\xi(n-1-i)} \quad ; \quad \xi \in \Xi \quad ; \quad i = 0, 1, \ldots, (k-1) . \tag{4-18}$$

On the other hand, the Lagrange interpolation formula [16,26] shows how each $p_i(X)$, degree $(n-k-1)$ or less, is completely specified by knowing the $(n-k)$ values given by equation (4-18).

$$p_i(X) = \sum_{\xi \in \Xi} W^{\xi(n-1-i)} \left[\frac{\prod_{\lambda \neq \xi} \left(X - W^{\lambda}\right)}{\prod_{\sigma \neq \xi} W^{\xi} - W^{\sigma}} \right]_{\lambda, \sigma \in \Xi} . \tag{4-19}$$

This interpolation result, when combined with equation (4-18), yields a closed form expression for the transform of the rows of G.

$$\hat{P}_{i,q} = W^{q(n-1-i)} - \sum_{\xi \in \Xi} W^{\xi(n-1-i)} \left[\frac{\prod_{\lambda \neq \xi} \left(W^{q} - W^{\lambda} \right)}{\prod_{\sigma \neq \xi} W^{\xi} - W^{\sigma}} \right]_{\lambda, \sigma \in \Xi} . \qquad (4\text{-}20)$$

The role of the root indexing subset Ξ is obvious now.

The inverse transform may be invoked to establish the following Parseval's type relationship expressing certain squared terms in E, equation (4-15).

$$\left| p_{k-1-i,r} \right|^2 = \frac{1}{n^2} \sum_{t=0}^{n-1} \sum_{q=0}^{n-1} W^{-r(t-q)} P_{k-1-i,t} \, \hat{P}^{*}_{k-1-i,q} . \qquad (4\text{-}21)$$

Then, performing the summation in equation (4-15) over index r leads to a result where further simplifications are possible because of the nonoverlapping nature of some summation indices.

$$E = \sum_{j=0}^{n-k-1} \sum_{i=0}^{k-1} \sigma_i^2 \sum_{\xi \in \Xi} \sum_{\zeta \in \Xi} W^{i(\xi-\zeta)} \omega(j,\xi) \, \omega^{*}(j,\zeta) \qquad (4\text{-}22a)$$

where

$$\omega(j,\xi) = \frac{1}{n} \sum_{t=0}^{n-1} W^{jt} W^{\xi(n-k)} \left[\frac{\prod_{\lambda \neq \xi} \left(W^{t} - W^{\lambda} \right)}{\prod_{\sigma \neq \xi} W^{\xi} - W^{\sigma}} \right]_{\lambda, \sigma \in \Xi} . \qquad (4\text{-}22b)$$

The noise power terms σ_i^2 are fixed by the noise statistics and are not affected by the code indexing subset Ξ. The summation index i in (4-22a) on some terms resembles a transform of the length k sequence $\left\{ \sigma_i^2 \right\}$, $i = 0, 1, \ldots, (k-1)$. The following is a legitimate transform sequence.

$$\hat{S}(r) = \sum_{i=0}^{k-1} \sigma_i^2 W^{ir} \quad ; \quad r = 0, 1, \ldots, (n-1) . \qquad (4\text{-}23)$$

Thus, the bilinear form E can be rewritten compactly.

$$E = \sum_{\xi \in \Xi} \sum_{\zeta \in \Xi} S(\xi - \zeta)\ \Omega_{\Xi}(\xi, \zeta) \tag{4-24a}$$

with the identification,

$$\Omega_{\Xi}(\xi, \zeta) = \sum_{j=0}^{n-k-1} \omega(j, \xi)\, \omega^{*}(j, \zeta). \tag{4-24b}$$

The function $\Omega_{\Xi}(\xi, \zeta)$ has $(n-k)^2$ values and there are n starting positions for the indexing subset Ξ. Thus it is possible to find the best indexing subset by straightforward calculations on the order of $n(n-k)^2$.

Modifying Real Convolutional Codes

Infinite sequence processing described by a finite difference equation (or equivalently, by a rational Z transform transfer function) will be protected by a systematic real convolutional code. The key to simplification lies in adjusting the parity calculation equations so as to cancel the poles of the transfer function, permitting computation in the parity estimation subassembly at a slower rate. The details of this novel approach follow.

The finite difference equation relating input to output samples is defined by two sets of coefficients.

$$v_r = \sum_{i=0}^{\nu} a_i\, u_{r-i} - \sum_{j=1}^{\delta} b_j\, v_{r-j}\,. \tag{4-25}$$

The corresponding transfer function $H(Z)$ can also be viewed as the ratio of the respective Z transforms of the input and output sequences. It is completely defined by the coefficients governing the difference equation.

$$\begin{aligned} &\text{INPUT} && \{u_i\} \leftrightarrow U(Z) \\ &\text{OUTPUT} && \{v_j\} \leftrightarrow V(Z) \end{aligned} \tag{4-26}$$

$$H(Z) = \frac{V(Z)}{U(Z)} = \frac{a_0 + a_1 Z^{-1} + a_2 Z^{-2} + \ldots a_\nu Z^{-\nu}}{b_0 + b_1 Z^{-1} + b_2 Z^{-2} + \ldots b_\delta Z^{-\delta}} \quad ; \quad b_0 = 1.$$

The inverse Z transform of $H(Z)$ yields the impulse response $\{h_j\}_{j=0}^{+\infty}$ that appears in matrix F, equation (3-30). For future reference, the numerator and denominator of this function will be identified notationally as

$$H(Z) = \frac{N(Z)}{D(Z)}. \tag{4-27}$$

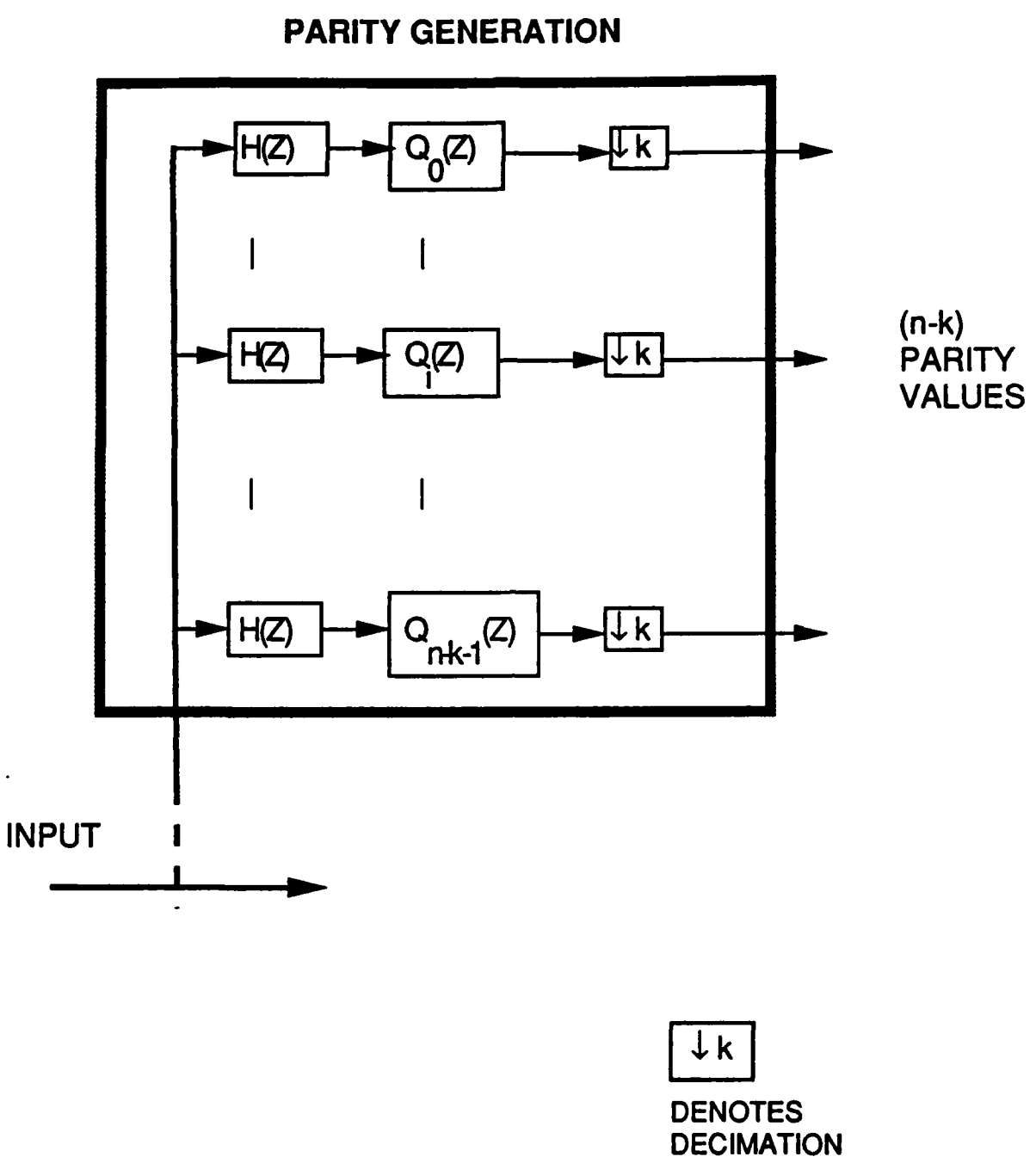
The composite filter described by the transfer function $H(Z)$ may be protected, in an inefficient way, by defining the parity generation and regeneration subassemblies shown in the respective parts of Figure 4-2. For the parity generation process, each parity channel in Figure 4-2a represents the combined effects of the transfer function $H(Z)$ and the respective parity weighting $\{q_c^{(i)}\}_{i=0}^{M-1}$, equation (2-9), defining the c^{th} parity position for the code. The Z transform of the c^{th} parity channel will be denoted by $Q_c(Z)$.

$$Q_c(Z) = \sum_{i=0}^{M-1} q_c^{(i)} Z^{-i} \quad ; \quad c = 0, 1, \ldots, (n - k - 1). \tag{4-28}$$

The decimator at the output of each channel is denoted by $\downarrow k$ indicating that only one sample is used for each k input samples [30].

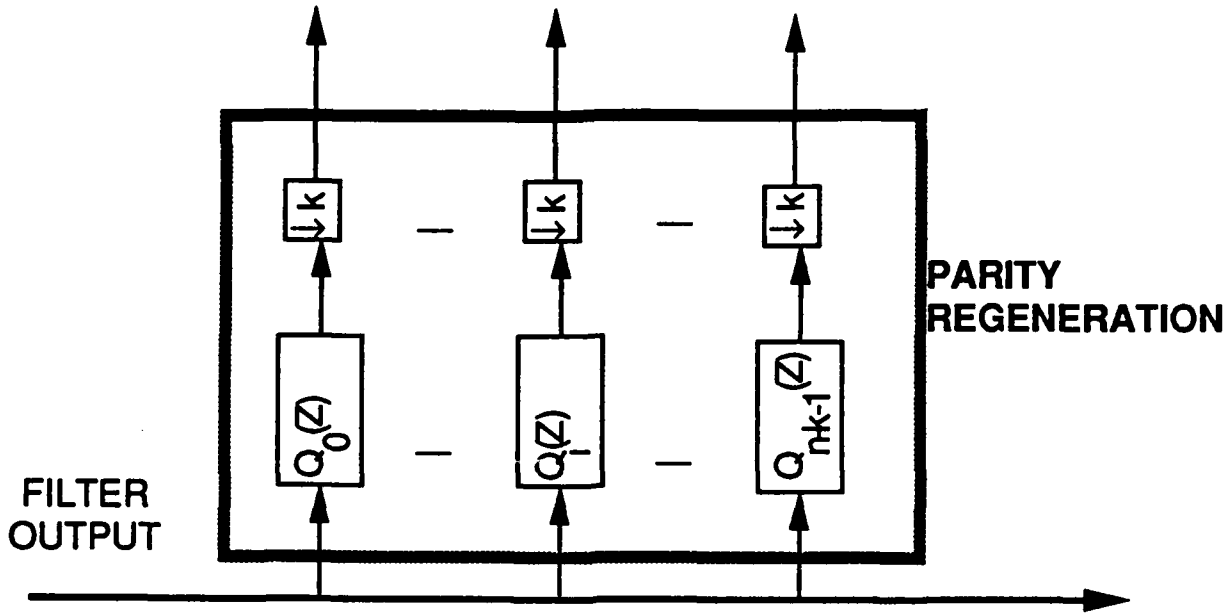
As was shown earlier, fault-tolerance is achieved by recomputing corresponding parity values from the filter system outputs. Any discrepancies, within the roundoff tolerance of the system and the error-detecting power of the code, indicates a protected malfunction somewhere in the system, including even the subsystems generating or regenerating the parity values. The parity estimation subassembly, outlined in Figure 4-2b, employs the usual FIR filter channels defined by the optimum estimator, equation (3-42), assuming all bias terms are zero.

Of course, this system configuration is grossly inefficient since it replicates $(n - k)$ actions of the original filter, one in each parity channel. Even the decimated sampling at reduced rate $\frac{1}{k}$ does not mitigate the increased complexity. However,



PARITY GENERATION
FIGURE 4 -2a

PARITY CALCULATIONS TO PROTECT TRANSFER FUNCTION WITH CONVOLUTIONAL CODE
FIGURE 4 -2



PARITY REGENERATION
FIGURE 4 -2b

PARITY CALCULATIONS TO PROTECT TRANSFER FUNCTION WITH CONVOLUTIONAL CODE
FIGURE 4 -2

other steps can drastically reduce this complexity bringing the implementation in the realm of practical applicability.

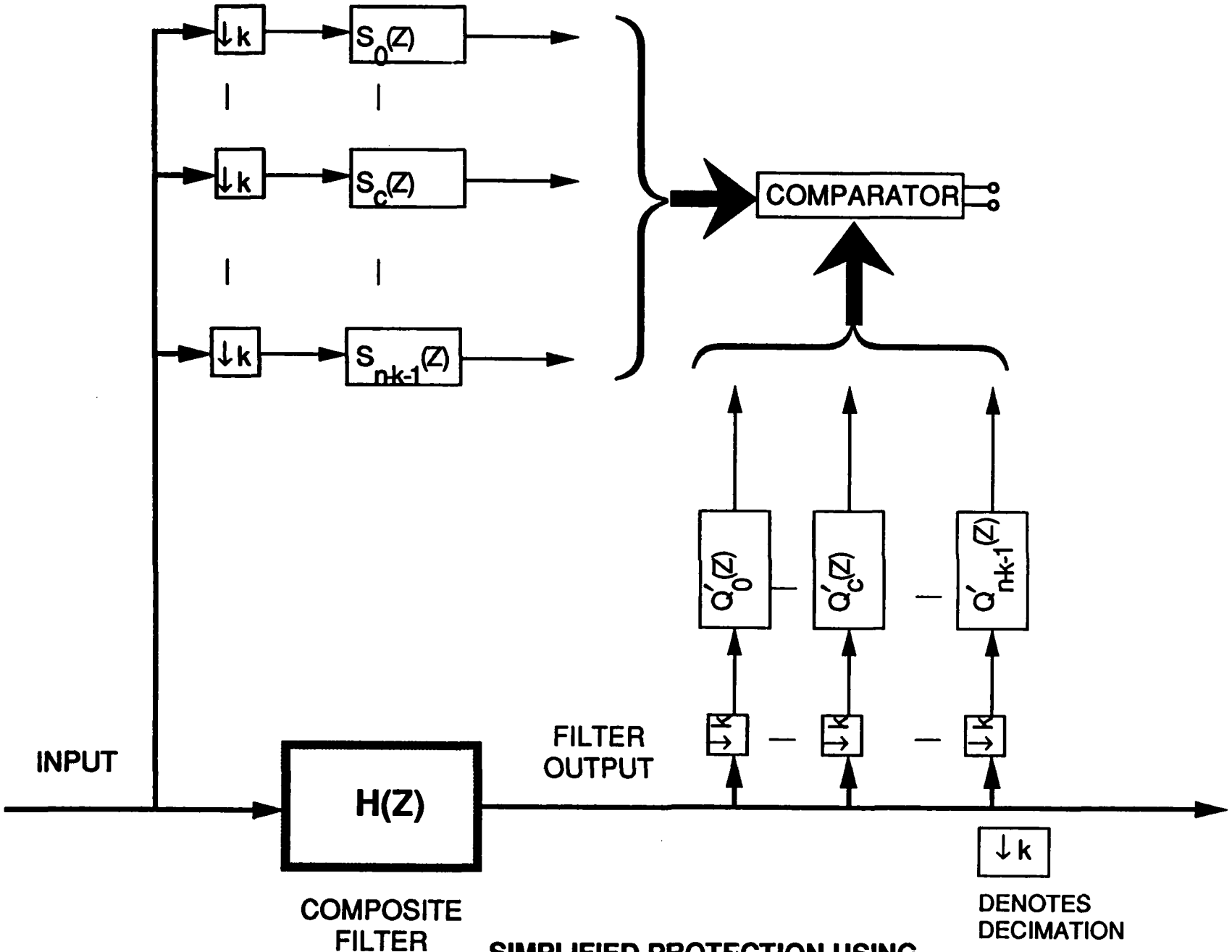
A framework for modifying rate $\frac{k}{n}$ convolutional codes while still preserving the code's original distance properties will be explored here. Algorithms for computing the distance of a convolutional code [15,31,32] examine the distance contribution resulting from each group of k input digits as mapped by a typical n columns of the generator matrix G, equation (2-6). The respective columns and rows affecting a generic n code digits are given by matrix $G^{(m)}$.

$$G^{(m)} = \begin{pmatrix} 0 & Q_m \\ 0 & Q_{m-1} \\ 0 & | \\ | & | \\ | & | \\ 0 & Q_1 \\ I_k & Q_0 \end{pmatrix} \quad (m+1)k \times n \text{ Code Segment Matrix.} \tag{4-29}$$

The right $(n-k)$ columns form the Q matrix, equation (2-8), which contains the quantities that are to be modified.

The minimum distance profile of a convolutional code relies on the interaction of k input digit groups with the row space of $G^{(m)}$. Hence, if a new matrix $G'^{(m)}$ containing a different Q' still has a combinatorially equivalent row space of the original $G^{(m)}$, equation (4-29) above, the minimum distance properties remain the same [15,31,32]. The rightmost columns Q' of the code segment defining the modified code $G'^{(m)}$ corresponds to $(n-k)$ new FIR parity channel filters designated $Q'_0(Z)$, $Q'_1(Z)$, ..., $Q'_{n-k-1}(Z)$. (See similar equations (2-8))

$$G'^{(m)} = \begin{pmatrix} 0 & \vdots \\ \cdots & \vdots \quad Q' \\ I_k & \vdots \end{pmatrix} \tag{4-30}$$



SIMPLIFIED PROTECTION USING MODIFIED CONVOLUTIONAL CODE

FIGURE 4 -3

However, if each of the new parity channel's transfer function contains $D(Z)$, the denominator of $H(Z)$, equation (4-27), the poles are effectively removed from the parity generation filters, and

$$Q'_c(Z) = D(Z)\,R_c(Z)$$

$$\textit{Implies} \tag{4-31}$$

$$H(Z)Q'_c(Z) = R_c(Z)\,N(Z) \triangleq S_c(Z)$$

$$c = 0, 1, 2, \ldots, (n-k-1)\,.$$

The parity channel only needs to implement the FIR filter described by the transfer function $S_c(Z)$. In addition, the decimation operation $\downarrow k$, commutes with such filter structures [30]. Figure 4-3 shows the theoretical impact of this simplification for typical channel $c = 0, 1, \ldots, (n-k-1)$. The theory achieving these simplifications will be developed next.

The columns of $G^{(m)}$ may be viewed as vectors from a complex vector space, V with dimension $[(m+1)k]$. Associated with every vector space is its dual space [24, Chapt. III, Section 6], a space of linear functionals from V into the complex numbers. The dual space of V is written as V^* and its members are also written as column vectors. The column space of $G^{(m)}$ is a subspace of V which is also uniquely specified by elements in the dual space. The subspace of V^* which annihilates the column space generated by $G^{(m)}$ is called the annihilator of this space, $Ann[G^{(m)}]$.

$$Ann[G^{(m)}] \subset V^*$$

$$\underline{\alpha} \in Ann[G^{(m)}] \leftarrow\rightarrow \underline{\alpha}^T G^{(m)} = 0 \tag{4-32}$$

$$\underline{\alpha}\ \textit{is}\ [(m+1)k]\ k \times 1\ \textit{Vector}$$

$$\alpha^T\ \textit{Denotes Transpose}$$

The matrix whose linearly independent columns span $Ann[G^{(m)}]$ is denoted by $H^{(m)}$.

In order to find the annihilator space generated by $H^{(m)}$, matrix $G^{(m)}$ is reduced by a series of column operations and row permutations to a canonical form for which the annihilator has an obviously simple configuration. The row permutations are represented by a nonsingular matrix D which in effect permutes the basis for V. Because of the limited type of operations represented by D, it has the property:

$$D^{-1} = D^T$$

The column operations are represented by the nonsingular $n \times n$ matrix C. The resultant canonical form for $G^{(m)}$ is given below where the I_{n-k} matrix is a consequence of the fact that $G^{(m)}$ is a mapping ONTO the n-tuples representing each codeword symbol.

$$DG^{(m)}C = \begin{pmatrix} 0 & S \\ 0 & I_{n-k} \\ I_k & 0 \end{pmatrix} \begin{matrix} C, n \times n \text{ Nonsingular} \\ D, (m+1)k \times (m+1)k \text{ Permutation} \end{matrix}$$

$$S \text{ is } ((m+1)k - n) \times (n-k) . \tag{4-33}$$

The matrix representing the annihilator of this canonical form is easily written:

$$(I_{(m+1)k-n} \mid -S \mid 0) \quad \textit{where submatrix } 0 \textit{ is } [(m+1)k - n] \times k.$$

However, in the original basis the annihilating matrix for $G^{(m)}$ is given by matrix, $H^{(m)T}$.

$$H^{(m)T} = (I_{(m+1)k-n} \mid -S \mid 0)\, D . \tag{4-34}$$

Then the usual dual space identity applies to any equivalent code segment generating matrix $J^{(m)}$.

$$H^{(m)T} J^{(m)} = 0 \qquad 0 \textit{ is } [(m+1)k - n] \times k \textit{ Zero Matrix} . \tag{4-35}$$

The columns of $H^{(m)}$ generate the column space of $Ann[G^{(m)}]$ because the matrix D^T is nonsingular due to its permutation origins.

$$Ann[G^{(m)}] = \textit{Column Space } H^{(m)}$$

$$H^{(m)} = D^T \begin{pmatrix} I_{(m+1)k-n} \\ -S^T \\ 0 \end{pmatrix} \quad 0 \text{ is } k \times [(m+1)k - n] . \tag{4-36}$$

A generator matrix with a different set of $(n-k)$ parity columns, Q' similar to those in equation (2-8) must satisfy the equality

$$H^{(m)T} \begin{pmatrix} 0 & \vdots & \\ \cdots & \vdots & Q' \\ I_k & \vdots & \end{pmatrix} = 0. \tag{4-37}$$

This guarantees that the columns of Q' generate a code segment. These equations can be separated into two parts with Q' appearing in only one equation.

$$H^{(m)T} Q' = 0. \tag{4-38}$$

The new columns in Q' must also eliminate the poles in $D(Z)$ to simplify each parity filter channel. This constraint may be expressed in terms of two unknown quantities R and Q' where R is the aggregate of the FIR filters described by transfer functions $R_c(Z)$, $c = 0, 1, \ldots, (n-k-1)$. On the other hand, a matrix A contains the effects of $D(Z)$.

$$A\,R = Q' \tag{4-39a}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ b_1 & 1 & 0 & - & - \\ b_2 & b_1 & | & | & | \\ b_3 & b_2 & - & - & - \\ - & - & - & 0 & 0 \\ - & - & - & 1 & 0 \\ - & - & - & b_1 & 1 \\ b_\delta & b_{\delta-1} & - & b_2 & b_1 \\ 0 & b_\delta & | & | & b_2 \\ 0 & 0 & - & - & | \\ | & | & | & b_\delta & - \\ 0 & 0 & 0 & 0 & b_\delta \end{pmatrix} \times \begin{pmatrix} R_{0,0} & R_{1,0} & - & R_{0,(n-k-1)} \\ R_{1,0} & - & - & R_{1,(n-k-1)} \\ R_{2,0} & - & - & - \\ - & | & - & - \\ - & . & | & - \\ . & | & . & . \\ | & - & | & | \\ . & - & - & . \\ R_{(M-\delta-1),0} & R_{(M-\delta-1),1} & - & R_{(M-\delta-1),(n-k-1)} \end{pmatrix} \tag{4-39b}$$

$$= \begin{pmatrix} Q'_{0,0} & Q'_{1,0} & - & Q'_{0,(n-k-1)} \\ Q'_{1,0} & - & - & Q'_{1,(n-k-1)} \\ Q'_{2,0} & - & - & Q'_{2,(n-k-1)} \\ - & | & | & - \\ - & . & . & - \\ . & | & | & . \\ | & - & - & | \\ . & - & - & . \\ Q'_{M-1,0} & Q'_{M-1,1} & - & Q'_{M-1,(n-k-1)} \end{pmatrix} \quad ; \quad M = (m+1)k .$$

The annihilator $H^{(m)T}$ reduces this to a homogeneous system when the additional constraints in equation (4-38) are imposed.

$$H^{(m)T} A R = 0. \tag{4-40}$$

This system will provide dependencies among the components of R.

The combined effects of the annihilator restriction and the poles of $H(Z)$ appear in the matrix product, giving the $[(m+1)k-n] \times [(m+1)k - \delta]$ matrix $\tilde{A}$.

$$\tilde{A} = H^{(m)T} A . \tag{4-41}$$

A series of row operations, represented by nonsingular matrix E brings $\tilde{A}$ to echelon form where the row rank is denoted by ρ.

$$E\tilde{A} = \begin{pmatrix} \tilde{A}_{sq} & | & B \\ - & . & - \\ 0 & | & 0 \end{pmatrix} \quad \begin{array}{l} \tilde{A}_{sq} \text{ is } \rho \times \rho \\ B \text{ is } \rho \times [(m+1)k - \delta - \rho] \end{array} \tag{4-42a}$$

$$\rho \leq \text{MIN}\{[(m+1)k-n], [(m+1)k-\delta]\} . \tag{4-42b}$$

Since E is nonsingular, the homogeneous equation (4-40) is equivalent to:

$$(\tilde{A}_{sq} \mid B)\, R = 0. \tag{4-43}$$

Constraints among components in the unknown matrix R follow from this equation. The components in R may be separated into independent and dependent parts following the partition of the matrix in equation (4-42a).

$$R = \left(\frac{R_D}{R_I}\right) \begin{matrix} \rho \times (n-k) \\ \textit{Dependent Part} \\ \hdots\hdots\hdots\hdots \\ [(m+1)k - \delta - \rho] \times (n-k) \\ \textit{Independent Part} \end{matrix} \tag{4-44}$$

The upper triangular matrix $\tilde{A}_{sq}$ is easily inverted, leading to the dependency between parts.

$$R_D = \tilde{A}_{sq}^{-1} B R_I . \tag{4-45}$$

Various choices for R_I provide potential solutions for the pole-canceling parity channels represented by Q'.

$$A\left(\frac{\tilde{A}_{sq}^{-1} B R_I}{R_I}\right) = Q' \quad ; \; Q' \text{ is } (m+1)k \times (n-k) . \tag{4-46}$$

The new choices for a code generating segment are contained in the matrix,

$$G^{(m)} = \begin{pmatrix} 0 & | \\ - & | \; A\left(\dfrac{-\tilde{A}_{sq}^{-1} B R_I}{R_I}\right) \\ I_k & | \end{pmatrix} . \tag{4-47}$$

There is no guarantee that any given code can be modified to cancel all the poles in a selected filter even if the respective dimensions are selected large enough. The row rank ρ of $\tilde{A}$, an $[(m+1)k - n] \times [(m+1)k - \delta]$ matrix determines the number of independent choices in R which in turn directly impacts the new filter weights in Q', a $(m+1)k \times (n-k)$ matrix which totally defines the code segment $G^{(m)}$ because of its systematic form.

A simple example employing one parity channel is given. The transfer function for a 7th order elliptic filter is

$$H(Z) = \frac{100(0.322928 - 0.382389Z^{-1} + 0.669801Z^{-2} - 0.061Z^{-3} - 0.062693Z^{-4} + 0.66983Z^{-5} - 0.381679Z^{-6} + 0.322016Z^{-7})}{1 - 5.116546Z^{-1} + 12.191418Z^{-2} - 17.263171Z^{-3} + 15.595655Z^{-4} - 8.963986Z^{-5} + 3.035586Z^{-6} - 0.468757Z^{-7}}$$

This filter will be protected with a rate 4/5 binary-based convolutional code taken from a table [33]. Its parity channel filter has Z transform $Q_c(Z)$.

$$Q_c(Z) = 1 + Z^{-1} + Z^{-2} + Z^{-3} + Z^{-4} + Z^{-5} + Z^{-8} + Z^{-10} + Z^{-11} + Z^{-12} + Z^{-15}$$

This is a systematic rate 4/5 code and has a minimum column distance function of 3. Thus, it can detect any three errors in a data block, where a block consists of four data words and one parity word.

After altering $Q(Z)$ to cancel the poles of $H(Z)$, the following $Q'_c(Z)$ is chosen:

$$Q'_c(Z) = 0.5631 + 0.3926Z^{-1} - 1.2983Z^{-2} - 0.7239Z^{-3} + 0.9201Z^{-4} + 1.5274Z^{-5}$$

$$- 0.8176Z^{-8} + 0.3971Z^{-10} - 1.4297Z^{-11} - 0.1094Z^{-12} + 0.5869Z^{-15}$$

The product of $H(Z)$ and $Q(Z)$ results in

$$S_c(Z) = 0.3491 + 0.5481Z^{-1} - 1.2378Z^{-2} + 0.9832Z^{-3} + 0.1847Z^{-4} - 0.2687Z^{-5}$$

$$+ 0.8345Z^{-6} + 0.2387Z^{-7} - 0.7845Z^{-8} + 0.9276Z^{-9} - 0.4218Z^{-10} - 0.1793Z^{-11}$$

$$+ 0.2947Z^{-12} - 1.3923Z^{-13} - 0.5782Z^{-14} + 0.8383Z^{-15}.$$

The impact of the code's parity filter responses on the MSE is quite complex. However, as equation (4-46) shows, there is a direct relationship between the choices of the independent part of the modified parity filter and the actual parity weighting values in Q'. This may be incorporated into the $\hat{\varepsilon}_f^2$ expression for the optimum MSE, equation (3-43). A typical term in $\hat{\varepsilon}_f^2$ containing a bilinear form in $\{q'^{(i)}\}$ reduces to a sum of squares when the noise components are assumed uncorrelated. This is analogous to the earlier case for cyclic codes, equations (4-13) through (4-15). A similar result for the case of a single parity channel has a typical term:

$$E_t = \sum_{i=0}^{M-1} \sigma_{t-i}^2 \left| q^{(i)} \right|^2. \qquad (4\text{-}48)$$

The individual noise powers are denoted by the σ_i^2 terms.

The influence of the code modification choices as expressed in equation (4-46) may be incorporated in this last equation. However, unlike the situation for cyclic codes, a discrete Fourier transform does not seem naturally applicable. In general, convolutional codes are not based directly on roots associated with the parity channel weighting. This is an important unsolved problem. Nevertheless, some form of transform may help decompose the problem further. There is an annihilator subspace affiliated with finding the modified parity channel weightings.

There are several other construction techniques for real convolutional codes besides the direct translation of binary codes to the real field. The rate 1/2 code constructed by Mathys [14] is one example. Another larger class of codes, to be described briefly below, is based on a technique due to Wyner [13, Sect. 13.3] and employs cyclic codes as a basis. The codes will be developed in their nonsystematic form for ease of exposition; it is a straightforward matter to transform them to a systematic format.

The Wyner real code has a free distance of six or more with redundancy of 3 positions ($n - k = 3$) and constraint parameter $m = 1$. The code length n is a design variable so that the rate is $(1 - 3/n)$. The subblocks of the generator matrix G, equation (2-6) containing a constraint length of the code, will be denoted by $G^{(1)}$. It in turn fully defines the two subblocks G_0 and G_1.

$$G^{(1)} = \begin{pmatrix} G_0 & G_1 \\ 0 & G_0 \end{pmatrix} \quad ; \quad 2k \times 2n \text{ Constraint Length Subblock}. \qquad (4\text{-}49)$$

The construction centers on the form of the parity-check matrix $H^{(1)}$ associated with $G^{(1)}$. This parity-check matrix contains two appropriately sized subblocks H_0 and H_1 that will be detailed fully in a moment.

$$H^{(1)} = \begin{pmatrix} \underline{1} & \underline{0} \\ H_0 & 0 \\ \underline{0} & \underline{1} \\ H_1 & H_0 \end{pmatrix} \quad ; \quad 2(n-k) \times H_i \text{ Submatrix} \tag{4-50a}$$

$\underline{1}$ Denotes 1 × n vector of all Ones

$\underline{0}$ Denotes 1 × n vector of all Zeros .

$$G^{(1)} H^{(1)h} = 0, \tag{4-50b}$$

where h Denotes the conjugate transpose.

The submatrices H_0 and H_1 are defined by a real cyclic code employing n^{th} complex roots of unity.

$$F^{(1)} = \begin{pmatrix} \underline{1} \\ H_0 \\ H_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ W^{(n-1)} & W^{(n-2)} & - & & W^2 & W^1 & 1 \\ W^{-(n-1)} & W^{-(n-2)} & & & W^{-2} & W^{-1} & 1 \\ \hline W^{2(n-1)} & W^{2(n-2)} & & & W^{+4} & W^{+2} & 1 \\ W^{-2(n-1)} & W^{-2(n-2)} & - & \cdots & W^{-4} & W^{-2} & 1 \end{pmatrix} \quad ; \quad W = \exp\left(\frac{j2\pi}{n}\right) \tag{4-51}$$

$$F^{(1)} = \begin{pmatrix} \underline{1} \\ H_0 \\ H_1 \end{pmatrix} \{5 \times n \text{ Matrix Defines Real BCH Code with } d_{min} = 6.\}$$

where

$$\begin{pmatrix} \underline{1} \\ H_0 \end{pmatrix} \left\{3 \times n \text{ Matrix Defines Code Containing } \left\{F^{(1)}\right\} \text{ with } d_{min} = 4.\right\}$$

The proof that this does have at least a minimum free distance of six may be found in Peterson and Weldon [13, Sect. 13.3]. The systematic form can be derived by applying row manipulations to $H^{(1)}$ resulting in corresponding part with a characteristic separation of parity part submatrices P_0 and P_1.

$$H_s^{(1)} = \begin{pmatrix} -P_0^T & I_3 & 0 & 0 \\ -P_1^T & 0 & -P_0^T & I_3 \end{pmatrix} \quad ; \quad P_i \; k \times (n-k) \text{ Matrix}$$

$$G_s^{(1)} = \begin{pmatrix} I_k & P_0 & 0 & P_1 \\ 0 & 0 & I_k & P_0 \end{pmatrix} \quad ; \quad k = n - 3 .$$

An example of the Q submatrix part of a systematic version of G can be generated by a simple computer program.

$$Q = \begin{pmatrix} +0.000000e+00 & -1.739739e-01 & +8.271757e-01 \\ +0.000000e+00 & -2.459447e+00 & -2.956523e+00 \\ +0.000000e+00 & -6.283361e+00 & -5.143227e+00 \\ +0.000000e+00 & -6.186813e+00 & -3.497567e+00 \\ +8.410501e-02 & -1.273087e+00 & +1.889822e+00 \\ +1.889822e-00 & -3.860601e+00 & +2.671619e+00 \\ +2.356570e-00 & -5.814091e+00 & +4.578434e+00 \\ +1.889822e-00 & -5.662539e+00 & +4.473557e+00 \end{pmatrix} .$$

$n = 7 \qquad k = 4 \qquad m = 1$

V. REFERENCES

[1] D. K. Pradhan, Editor, *Fault-Tolerant Computing Theory and Techniques, Volume I.* Englewood Cliffs: Prentice-Hall, 1986.

[2] K.-H. Huang and J. A. Abraham, "Algorithm-based Fault Tolerance for Matrix Operations," *IEEE Transactions on Computers*, Vol. C-33, pp. 518-528, 1984.

[3] J.-Y. Jou and J. A. Abraham, "Fault-Tolerant Matrix Arithmetic and Signal Processing on Highly Concurrent Computing Structures," *Proceedings of the IEEE* (Special Issue on Fault Tolerance in VLSI), Vol. 74, pp. 732-741, 1986.

[4] J. A. Abraham, "Fault Tolerance Techniques for Highly Parallel Signal Processing Architectures," *SPIE Highly Parallel Signal Processing Architectures*, Vol. 614, pp. 49-65, 1986 (K. Bromley, Editor).

[5] C. J. Anfinson and F. T. Luk, "A Linear Algebraic Model of Algorithm-Based Fault Tolerance," *IEEE Transactions on Computers*, Vol. C-37, pp. 1599-1604, 1988.

[6] F. T. Luk and H. Park, "An Analysis of Algorithm-Based Fault Tolerance Techniques," *SPIE Advanced Algorithms and Architectures for Signal Processing*, Vol. 696, pp. 222-227, 1986.

[7] W. G. Bliss, "Area-Time Efficient and Fault-Tolerant VLSI Arrays for Digital Signal Processing," Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Colorado, 1988.

[8] C. J. Anfinson, R. P. Brent and F. T. Luk, "A Theoretical Foundation for the Weighted Checksum Scheme, *SPIE Advanced Algorithms and Architectures for Signal Processing*, Vol. 975, pp. 10-18, 1988.

[9] R. P. Brent, F. T. Luk and C. J. Anfinson, "Choosing Small Weights for Multiple Error Detection," *SPIE High Speed Computing*, Vol. 1058, pp. 16-1—16-7, 1989.

[10] F. T. Luk, "Algorithm-Based Fault Tolerance for Parallel Matrix Equation Solvers," *SPIE Real-Time Signal Processing*, Vol. 564, pp. 49-53, 1985 (W. J. Miceli and K. Bromley, Editors).

[11] T. G. Marshall, Jr., "Coding of Real-Number Sequences for Error Correction: A Digital Signal Processing Problem," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, pp. 381-392, 1984.

[12] J. K. Wolf, "Redundancy, the Discrete Fourier Transform, and Impulse Noise Cancellation," *IEEE Transactions on Communications*, Vol. COM-31, pp. 458-461, 1983.

[13] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes (Second Edition)*, Cambridge, Massachusetts: The MIT Press, 1972.

[14] P. Mathys, "Rate $\frac{1}{2}$ Convolutional Codes Over the Reals," Private Communication (and part of Class Notes, University of Colorado), 1988.

[15] S. Lin and D. J. Costello, Jr., *Error Control Coding Fundamentals and Applications.* Engelwood Cliffs: Prentice-Hall, 1983.

[16] N. K. Bose, *Digital Filters Theory and Applications.* New York: North-Holland, 1985.

[17] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs: Prentice-Hall, 1989.

[18] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing.* New York: Macmillan, 1988.

[19] S. A. Tretter, *Introduction to Discrete-Time Signal Processing.* New York: Wiley, 1976.

[20] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, Second Edition. New York: McGraw-Hill, 1984.

[21] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding.* New York: McGraw-Hill, 1979.

[22] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part I.* New York: John Wiley and Sons, 1968.

[23] J. Wakerly, *Error Detecting Codes, Self-Checking Circuits and Applications.* New York: North-Holland, 1978.

[24] S. Lang, *Algebra (Second Edition).* Menlo Park, California: Addison-Wesley, 1984.

[25] W. B. Davenport, Jr., *Probability and Random Processes.* New York: McGraw-Hill, 1970.

[26] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Reading, Massachusetts: Addison-Wesley Publishing Co., 1985.

[27] T. G. Marshall, Jr., "Real Number Transform and Convolutional Codes," *Proceedings 24th Midwest Symposium on Circuits and Systems*, Albuquerque, NM, pp. 650-653, June 1981.

[28] J. K. Wolf, "Redundancy and the Discrete Fourier Transform," *Proceedings of the 1982 Conference on Information Sciences and Systems*, pp. 156-158, 1982.

[29] G. R. Redinbo, "System Level Reliability in Convolution Computations," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-37, pp. 1241-1252, 1989.

[30] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Englewood Cliffs: Prentice-Hall, 1983.

[31] K. J. Larsen, "Comments on 'An Efficient Algorithm for Computing Free Distance'," *IEEE Transactions on Information Theory*, Vol. IT-19, pp. 577-579, 1973.

[32] L. R. Bahl, C. D. Cullum, W. D. Frazer, and F. Jelinek, "An Efficient Algorithm for Computing Free Distance," *IEEE Transactions on Information Theory*, Vol. IT-18, pp. 437-439, 1972.

[33] J. Hagenauer, "High Rate Convolutional Codes with Good Distance Profiles," *IEEE Transactions on Information Theory*, Vol. IT-23, pp. 615-618, 1977.

[34] R. E. Blahut, "Algebraic Fields, Signal Processing and Error Control," *Proceedings of the IEEE*, Vol. 73, pp. 874-893, 1985.

[35] J. H. Cozens and L. A. Finkelstein, "Computing the Discrete Fourier Transform Using Residue Number Systems in a Ring of Algebraic Integers," *IEEE Transactions on Information Theory*, Vol. IT-31, pp. 580-588, 1985.

[36] S. S. Guillory, J. A. Martin, G. R. Redinbo and B. G. Zagar, "Fault-Tolerant Design Methods of VLSI Digital Filter Implementations," *VLSI Signal Processing III*, Chapter 35. New York: IEEE Press, 1989 (H. S. Moscovitz and R. W. Brodersen, Editors).

[37] G. R. Redinbo and B. G. Zagar, "Watchdog Parity Channels for Digital Filter Protection," *Proceedings of the Eighteenth International Symposium on Fault-Tolerant Computing*, Vol. FTCS-18, pp. 186-193, 1988.

[38] S. S. Guillory, "Concurrent Error-Detection in Digital Filters Using Convolutional Codes," M.S. Thesis, Department of Electrical Engineering and Computer Science, University of California, Davis, 1989.